zakboekje voor de

instructiesets
tabellen
conversielijsten

wessel akkermans

kluwer technische boeken

Sinclair ZX81 Personal Computer

GUARANTEE

Your Sinclair ZX81 is covered by a 12-month, comprehensive guarantee - valid in the UK only.

RETURNS/INFORMATION

If you need to return your computer for any reason, please send it (together with mains adaptor and all leads) plus relevant correspondence to -

Sinclair Research (JBS)
Stanhope Road
Camberley
Surrey GU15 3PS

If you have assembled your computer from a kit, please enclose a service fee of £10 - which will be returned to you if defective components are diagnosed.

For all other information, please write to the above address or telephone (0276) 21282.

sinclair

C.J. van Dorp

Zakboekje voor de ZX81

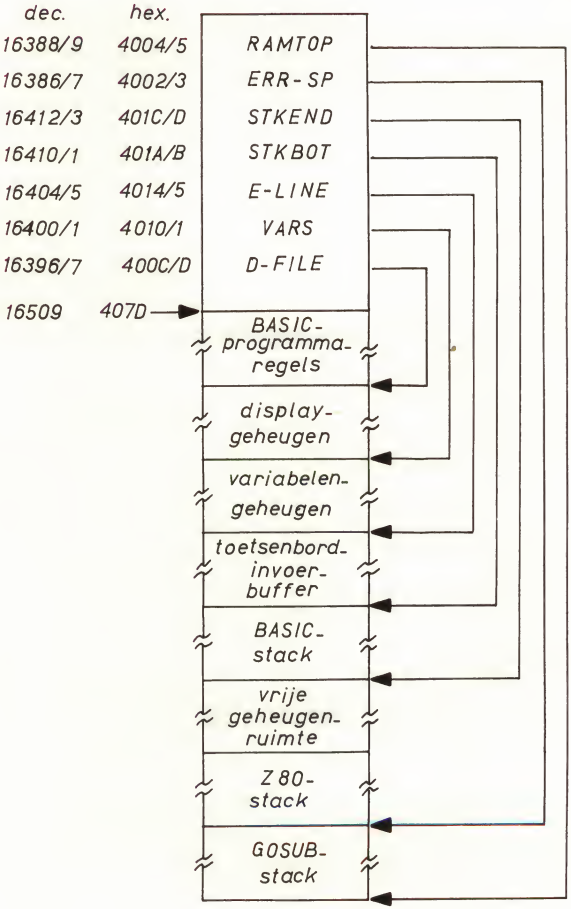
Wessel Akkermans

Zakboekje voor de ZX81

Instructiesets
tabellen
conversielijsten

kluwer technische boeken b.v.– deventer-antwerpen

ZX 81 geheugenindeling, algemeen



Systeemvariabelen

adres dec	adres hex	systeemvariabele naam	betekenis/functie
16384	4000	ERR.NR	Bevat de code van systeemboodschappen minus de waarde 1.
16385	4001	FLAGS	Bevat diverse controle-indicatoren ten behoeve van de BASIC-interpreter.
16386/7	4002/3	ERR.SP	De GOSUB-stackpointer.
16388/9	4004/5	RAMTOP	Het hoogste RAM-adres +1 dat voor BASIC beschikbaar is.
16390	4006	MODE	Momentele cursor-mode.
16391/2	4007/8	PPC	Regelnummer van de in uitvoering zijnde statement.
16393	4009	VERSN	Een waarde 0 hierin geeft aan dat het met SAVE weggeschreven programma BASIC voor ZX81 bevat.
16394/5	400A/B	E.PPC	Regelnummer van de regel die het cursor-teken bevat en die dus bij EDIT zal worden geselecteerd.
16396/7	400C/D	D.FILE	Het adres van het begin van het beeldschermgeheugen.
16398/9	400E/F	DF.CC	Het adres van de momentele printpositie binnen het beeldschermgeheugen.
16400/1	4010/1	VARS	Het adres van het begin van het variabelengeheugen.
16402/3	4012/3	DEST	Het adres van de variabele (binnen het variabelengeheugen) waaraan momenteel een waarde wordt toegekend.
16404/5	4014/5	E.LINE	Startadres van de EDIT-regel.
16406/7	4016/7	CH.ADD	Het adres van het volgende te interpreteren karakter.
16408/9	4018/9	X.PTR	Het adres van het karakter dat een syntax-fout veroorzaakt.
16410/1	401A/B	STKBOT	Het startadres van de stack voor de rekeneenheid.

adres dec	adres hex	systeemvariabele naam	betekenis/functie
16412/3	401C/D	STKEND	Het laatste adres van de stack voor de rekeneenheid.
16414	401E	BREG	Het B-register van de rekeneenheid.
16415/6	401F/20	MEM	Startadres van het deel van het geheugen dat door de rekeneenheid kan worden gebruikt. Dit wijst meestal naar MEMBOT.
16418	4022	DF.SZ	Aantal regels waaruit het onderste deel van het beeldscherm bestaat. Meestal is dit 2.
16419/20	4023/4	S.TOP	Regelnummer van de bovenste regel op het scherm.
16421/2	4025/6	LAST.K	Geeft het laatst ingetypte karakter aan.
16423	4027	DB.ST	De debounce-status van het toetsenbord.
16424	4028	MARGIN	Het aantal lege regels aan de onder- of bovenkant van het beeldscherm.
16425/6	4029/A	NXTLIN	Het adres van de volgende programmaregel die moet worden uitgevoerd.
16427/8	402B/C	OLDPPC	Het adres van de programmaregel die moet worden uitgevoerd op het commando CONT.
16429	402D	FLAGX	Diverse indicatoren.
16430/1	402E/F	STRLEN	Lengte van de string waaraan een waarde wordt toegekend.
16432/3	4030/1	T.ADDR	Het adres van het volgende item in de syntax-tabel.
16434/5	4032/3	SEED	De variabele van RAND, die zal worden gebruikt bij het bepalen van het random-getal met RND.
16436/7	4034/5	FRAMES	Het aantal gegenereerde TV-beelden.
16438/9	4036/7	COORDS	Coördinaten van het laatst geplotte pixel.
16440	4038	PR.CC	Het minst significante deel van het adres van de LPRINT-positie in het printbuffer.
16441/2	4039/A	S.POSN	Coördinaten van het laatste met PRINT AT geprinte karakter.

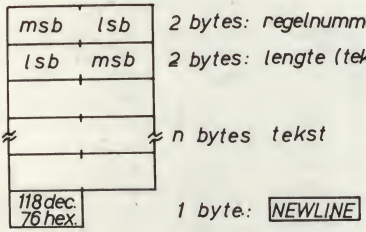
adres dec	adres hex	systeemvariabele naam	betekenis/functie
16443	403B	CDFLAG	De FAST/SLOW-indicator.
16444/76	403C/5C	PRBUFF	Het buffer voor de LPRINT-data.
16477/06	405D/7A	MEMBOT	Pointer naar het rekengeheugen.
16509	407D		Op dit adres start het programmageheugen.

In de hierna volgende tabel worden de systeemvariabelen nogmaals gegeven, echter nu in alfabetische volgorde van hun namen.

naam	dec	hex	naam	dec	hex
BREG	16414	401E	MEMBOT	16477/506	405D/7A
CDFLAG	16443	403B	MODE	16390	4006
CH.ADD	16406/7	4016/7	NXTLIN	16425/6	4029/A
COORDS	16438/9	4036/7	OLDPPC	16427/8	402B/C
DB.ST	16423	4027	PRBUFF	16444/76	403C/5C
DEST	16402/3	4012/3	PPC	16391/2	4007/8
DF.CC	16398/9	400E/F	PR.CC	16440	4038
DF.SZ	16418	4022	RAMTOP	16388/9	4004/5
D.FILE	16396/7	400C/D	SEED	16434/5	4032/3
ERR.NR	16384	4000	STKBOT	16410/1	401A/B
ERR.SP	16386/7	4002/3	STKEND	16412/3	401C/D
E.LINE	16404/5	4014/5	STRLEN	16430/1	402E/F
E.PPC	16394/5	400A/B	S.POSN	16441/2	4039/A
FLAGS	16385	4001	S.TOP	16419/20	4023/4
FLAGX	16429	402D	T.ADDR	16432/3	4030/1
FRAMES	16436/7	4034/5	VARS	16400/1	4010/1
LAST.K	16421/2	4025/6	VERSN	16393	4009
MARGIN	16424	4028	X.PTR	16408/9	4018/9
MEM	16415/6	401F/20			

Programmageheugen

Het programmageheugen begint op adres 16509 (decimaal), ofwel adres 407D (hexadecimaal). De lengte is variabel en wordt bepaald door de lengte van het programma.
Het programma bestaat uit regels en die zijn dan ook bepalend voor de verdere indeling van het programmageheugen.
Alle programmaregels worden direct achter elkaar in het programmageheugen opgeslagen, waarbij iedere regel een verdere onderverdeling kent, zoals is aangegeven in de hierna volgende afbeelding



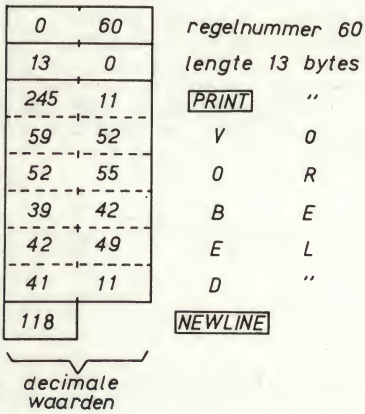
msb betekent: meest significant byte
lsb betekent: minst significant byte

Het regelnummer is het door u in het programma aangegeven regelnummer.
Op de volgende bladzijde wordt nog een voorbeeld van een programmaregel gegeven.

In onderstaande afbeelding is de volgende programmaregel afgebeeld:

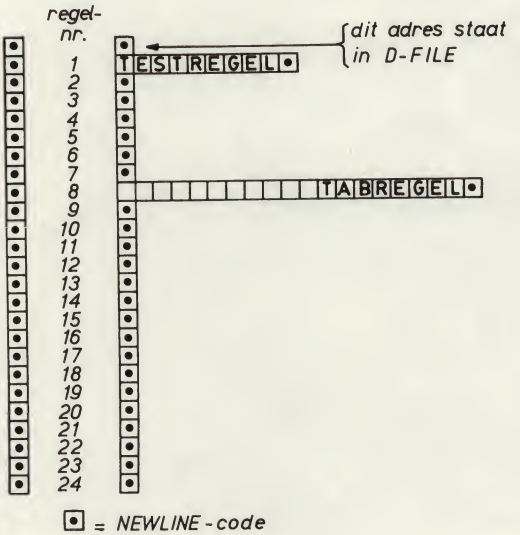
```
60 PRINT "VOORBEELD"
```

Merk op dat het lengteveld de lengte geeft van alle bytes die na het lengteveld nog in de regel staan. Inclusief het NEWLINE-karakter, doch exclusief het regelnummer en het lengteveld.
Ook is hier duidelijk te zien dat de significantie van de bytes in het regelnummerveld en in het lengteveld precies tegenovergesteld is. Bij het gebruik van deze velden in machinetaalprogramma's of in PEEK- en POKE-instructies dient u hiermee rekening te houden.



Beeldschermgeheugen

Het beeldschermgeheugen begint op het adres dat is aangegeven in de systeemvariabele D-FILE.
De lengte is variabel en wordt bepaald door het aantal karakters dat op het beeldscherm wordt afgedrukt.
Een geheel leeg beeldscherm zou in het geheugen alleen 25 NEWLINE-karakters te zien geven. In de afbeelding is dit aan de linker kant weergegeven. U ziet dat het geheugen begint met een NEWLINE-karakter en dat er vervolgens voor iedere regel een NEWLINE-karakter in staat.
Wordt er nu naar het beeldscherm geschreven, dan zal de tekst in de beschreven regel vóór het NEWLINE-karakter komen te staan. Ook spaties gelden als tekst en nemen dus geheugenruimte in.



In de afbeelding op de vorige pagina zijn twee voorbeelden gegeven. Zie hiervoor in de afbeelding de situatie die aan de rechter kant is weergegeven.
Met behulp van de statement:

```
PRINT "TESTREGEL"
```

is regel 1 van het beeldschermgeheugen beschreven.
Met behulp van de statement:

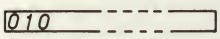
```
PRINT AT 8,12;"TABREGEL"
```

is regel 8 van het beeldscherm beschreven.
Zoals uit het voorgaande blijkt, kan het (zeker voor 1K RAM) erg belangrijk zijn alle tekst zoveel mogelijk aan de linker kant van het beeldscherm te zetten, om zodoende geheugenruimte te sparen.
In 16K geheugens zal echter het beeldschermgeheugen altijd de maximale grootte hebben. In iedere regel staan dan al direct na het aanschakelen 32 spaties plus een NEWLINE-karakter. In deze situatie is er dan ook geen enkele reden om tekst aan de linker kant van het beeldscherm te houden.

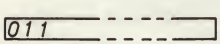
Opmerking:
Ingeval u met behulp van de systeemvariabele RAMTOP de voor BASIC beschikbare geheugenruimte kleiner maakt dan 3¼K, dan zal het beeldschermgeheugen weer variabel van lengte worden.

Variabelengeheugen

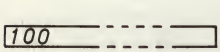
Het variabelengeheugen begint op het adres dat is aangegeven in de systeemvariabele VARS.
De lengte van het variabelengeheugen is variabel en hangt af van het aantal variabelen dat er in is opgenomen en de lengte van ieder van die variabelen.
Er zijn 6 verschillende soorten variabelen.
De eerste drie bits van iedere variabele geven de soort aan. Deze soorten zijn hieronder weergegeven:



stringvariabele (LET A$="ABC")



1-letter numerieke variabele (LET A=123)



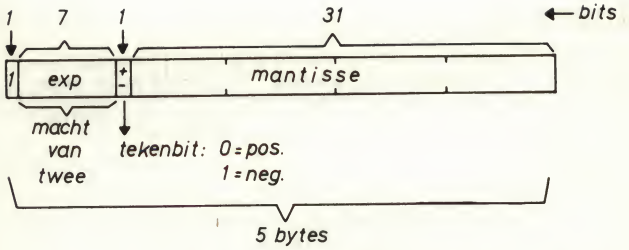
numerieke array (DIM A(4,5))

meerletter numerieke variabele (LET ABC=123)

alfanumerieke array (DIM A$(4,5))

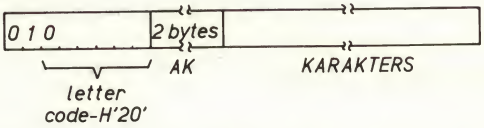
control-variabele (FOR A=1 TO 123 NEXT A)

In alle numerieke variabelen worden de waarden door middel van een veld van 5 bytes aangegeven. Dit 5 bytes veld heeft een formaat als in de volgende afbeelding.



In de formaten van de hierna volgende numerieke variabelen zullen de numerieke waarden worden aangegeven met "5 bytes". Op volgorde van de eerste drie bits van iedere variabele volgen nu de verschillende formaten.

stringvariabele

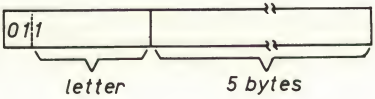


letter/code-H'20' - Daar het derde bit van links 0 is, is van de code van de stringnaam (letter) hexadecimaal 20 afgetrokken. Om de werkelijke stringnaam terug te krijgen moet men dus H'20' bij de gevonden code optellen.

AK - In dit twee bytes veld wordt de lengte van het tekstveld (karakters) gegeven.

KARAKTERS - Dit zijn de karakters van de gespecificeerde string.

1-letter numerieke variabele



5 bytes - Het formaat van dit veld is hierboven aangegeven.

ZX 81 BASIC-instructies

In de hiernavolgende tabel van BASIC-instructies en commando's zal voor iedere BASIC-instructie een zogenaamde syntaxomschrijving worden gegeven. In deze syntaxomschrijving worden een aantal tekens gebruikt waarvan de betekenis hier volgt:

[ ]	mag geheel worden weggelaten.
⌊ ⌋	één van de door | van elkaar gescheiden termen moet worden gebruikt.
|	scheiding van termen.
$_{1}^{n}\{$ $\}$	term of aantal termen die meer dan éénmaal mag worden ingegeven.
< >	omschrijving.

instructie	syntax/omschrijving
CLEAR	CLEAR Wist het variabelengeheugen.
CLS	CLS Wist het beeldschermgeheugen.
CONT	CONT Zet het programma voort met dezelfde regel indien de stopcode ≠ 9, met de volgende regel indien de stopcode = 9.
COPY	COPY Kopiëert het beeldschermgeheugen naar de printer indien deze is aangesloten. Indien COPY als commando wordt gegeven dan wordt het beeldscherm niet gewist voordat het commando wordt uitgevoerd.

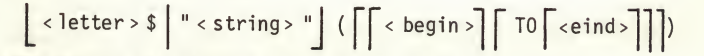
instructie	syntax/omschrijving
DIM	DIM <letter> (<array-index> [1 n {, <array-index> }]) Creëert ruimte in het variabelengeheugen voor een numerieke array met een grootte die van de opgegeven dimensies afhangt.
DIM	DIM <letter>$(<array-index> [1 n {, <array-index> }]) Creëert ruimte in het variabelengeheugen voor een string-array met een grootte die van de opgegeven dimensies afhangt.
FAST	FAST Hierna genereert de computer alleen TV-beelden als hij geen instructies uitvoert.
FOR..TO.. ..STEP	FOR <letter> = <nummer> TO <nummer> [STEP <nummer>] <letter> is de variabelennaam. <nummer> is een numerieke uitdrukking. De variabele krijgt de waarde van het daarachterstaande nummer. Hierna worden de volgende instructies uitgevoerd, tot de NEXT-instructie wordt uitgevoerd. De variabele krijgt nu de waarde die het al had + of - de waarde van het nummer achter STEP. Hierna worden weer alle instructies tot de NEXT uitgevoerd, enz.. Dit gaat door tot de variabele de waarde achter TO heeft bereikt. Indien STEP wordt weggelaten dan neemt het BASIC-systeem aan dat de stapgrootte +1 is.
GOSUB	GOSUB <regelnummer> Springt naar het aangegeven regelnummer, doch onthoudt het volgende regelnummer. Zodra een RETURN-instructie wordt uitgevoerd, wordt naar het onthouden regelnummer teruggesprongen.
GOTO	GOTO <regelnummer> Springt naar het aangegeven regelnummer.

instructie	syntax/omschrijving
IF..THEN..	IF <uitdrukking> THEN <instructie> Indien de uitdrukking waar is, dan wordt de achter THEN staande instructie uitgevoerd.
INPUT	INPUT <variabele> [$] Invoer van numerieke of string gegevens. Het $-teken moet alleen worden gebruikt wanneer met de INPUT karakters moeten worden ingevoerd in een stringvariabele.
LET	LET <variabele> [$] = <uitdrukking> Kent de numerieke of alfanumerieke waarde van de uitdrukking toe aan de variabele. Het $-teken moet alleen worden gebruikt wanneer karakters moeten worden toegekend aan een stringvariabele.
LIST	LIST [<regelnummer>] Drukt programmaregels af op het beeldscherm beginnende bij programmaregel <regelnummer>. Indien regelnummer wordt weggelaten dan worden alle programmaregels vanaf het begin van het programma afgedrukt.
LLIST	LLIST [<regelnummer>] Als LIST, doch nu wordt er afgedrukt op de printer.
LOAD	LOAD "[<programmanaam>]" Leest het programma met de naam <programmanaam> van cassette. Indien geen programmanaam is opgegeven dan wordt het eerstvolgende programma van cassette gelezen.
LPRINT	LPRINT [$_1^n${[<printitem>]$_1^n${[, | ;]}}] Indien het hele argument wordt weggelaten dan heeft PRINT tot gevolg dat de volgende PRINT op de volgende regel zal afdrukken. Indien alleen een ; als argument wordt gebruikt dan heeft de instructie geen gevolg. Indien alleen een , wordt gebruikt als argument dan zal de volgende PRINT in de volgende kolom afdrukken, dit kan dus ook een volgende regel zijn.

instructie	syntax/omschrijving
LPRINT (vervolg)	Indien alleen een printitem als argument wordt gebruikt, dan zal de volgende PRINT-instructie op de volgende regel afdrukken. Meerdere printitems moeten altijd door een komma of puntkomma van elkaar gescheiden zijn. LPRINT drukt af op de printer. Puntkomma heeft "geen" verschuiving van de printpositie tot gevolg. Komma heeft verschuiving van de printpositie naar de volgende kolom tot gevolg.
NEW	NEW Het RAM wordt tot aan het in RAMTOP aangegeven adres gewist en getest.
NEXT	NEXT <letter> Veroorzaakt een sprong naar de bijbehorende FOR-instructie en verhoogt of verlaagt de met letter aangegeven variabele.
PAUSE	PAUSE < numerieke uitdrukking > De numerieke uitdrukking geeft aan hoeveel keer er 1/50 seconde moet worden gewacht. De numerieke uitdrukking mag wel nul zijn, maar mag niet worden weggelaten.
PLOT	PLOT < hor.pos. > , < vert.pos. > Het beeldscherm is opgedeeld in 64 horizontale en 44 verticale punten (zogenaamde pixels) die met 0 tot en met 63 respectievelijk met 0 tot en met 43 kunnen worden aangeduid. PLOT maakt het aangegeven pixel zwart.
PRINT	PRINT $\left[\begin{smallmatrix}n\\1\end{smallmatrix}\left\{\left[\begin{smallmatrix}\text{< printitem >}\end{smallmatrix}\right]\begin{smallmatrix}n\\1\end{smallmatrix}\left\{\begin{smallmatrix},\\;\end{smallmatrix}\right\}\right\}\right]$ Werkt als LPRINT, doch nu op beeldscherm.
RAND	RAND [< numerieke uitdrukking >] De numerieke uitdrukking wordt gebruikt voor het bepalen van het begin van de rij willekeurige getallen. Indien hij wordt weggelaten of de waarde nul heeft, wordt de TV-beeldenteller gebruikt voor het bepalen van het begin van de rij willekeurige getallen. Dit laatste is het meest willekeurig.

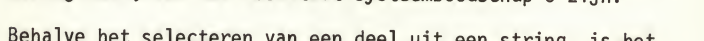
instructie	syntax/omschrijving
REM	REM ⌈ < tekst > ⌉ Alle achter de REM opgenomen tekst wordt als commentaar beschouwd. Deze informatie wordt niet door het programma gebruikt. Wel kan men deze statement gebruiken om er machinetaal-routines in op te slaan.
RETURN	RETURN Gaat terug naar de programmaregel onmiddellijk volgend op de bijbehorende GOSUB-instructie.
RUN	RUN ⌈ < regelnummer > ⌉ Start het programma op de aangegeven regel. Indien geen regelnummer werd gespecificeerd, dan wordt het programma op de eerste regel gestart. Alvorens het programma te starten wordt eerst het variabelengeheugen gewist.
SAVE	SAVE " < programmanaam > " Schrijft het aangegeven programma naar cassette. Met LOAD kan het programma onder dezelfde naam weer worden teruggelezen. Denk er wel aan de cassetterecorder eerst te starten in de stand opname.
SCROLL	SCROLL Schuift het hele beeldscherm 1 regel omhoog en voegt een lege nieuwe lijn toe aan de onderkant.
SLOW	SLOW Zet de computer in een toestand waarin alleen tussen het genereren van TV-beelden door instructies worden uitgevoerd.
STOP	STOP Stopt de uitvoering van het programma met een systeemboodschap (code 9).
UNPLOT	UNPLOT < hor.pos. > , < vert.pos. > Werkt als PLOT, maar wist nu de met PLOT zwart gemaakte "pixel".

Slicing is geen BASIC-instructie, doch het is een ZX81 manier om de normale BASIC-instructies MID$, LEFT$ en RIGHT$ te simuleren.
De syntax is als volgt:



Met behulp van slicing kan men een deel van een tekst selecteren. Deze tekst kan ofwel direct in een string zijn opgenomen of in een stringvariabele staan.
Het gewenste tekstdeel (slice) kan uit een string worden geselecteerd met behulp van een begin- en eindnummer.
Indien < begin > wordt weggelaten dan gaat de "slice" van het begin van de string tot aan het met < eind > aangegeven karakter.
Indien < eind > wordt weggelaten dan gaat de "slice" vanaf het met <begin> aangegeven karakter tot en met het einde van de string.
Indien de hele term tussen de haakjes wordt weggelaten, dan wordt de gehele string geselecteerd.
Indien alleen een cijfer tussen de haakjes staat, dan wordt alleen het met dat cijfer aangegeven karakter uit de string geselecteerd.
Indien < begin > groter is dan < eind > , dan zal het resultaat een lege string zijn ("").
Indien één van de waarden <begin> of <eind> buiten de string valt, zal het resultaat systeemboodschap 3 zijn.

Behalve het selecteren van een deel uit een string, is het ook nog mogelijk een aantal strings tot één string samen te voegen. De syntax hiervoor is als volgt:



Dit samenvoegen van strings (concatination) levert een zogenaamde gecompliceerde string op.
Ook op gecompliceerde strings is slicing mogelijk.
Slicing heeft echter een hogere prioriteit dan samenvoegen, zodat u het gebruik van haakjes goed in de gaten dient te houden.

functienaam	functie
PEEK n	Geeft de inhoud van geheugenadres n. De maximum waarde van n = 65535.
PI	Vertegenwoordigt π = 3,14159265
RND	Geeft een "pseudo-random"-nummer. De waarde van dit nummer kan variëren van 0 tot 1. RND genereert 65536 verschillende nummers die steeds weer in de zelfde volgorde worden gegenereerd. Met de BASIC-statement RAND n kan men de plaats waar RND de reeks getallen start beïnvloeden. Zie daarvoor de RAND-instructie in het hoofdstuk ZX81 BASIC instructies en commando's.
SGN n	Geeft tekeninformatie over het getal n. Indien n = negatief, dan is SGN n = -1 Indien n = nul, dan is SGN n = 0 Indien n = positief, dan is SGN n = +1
SIN n	Geeft de sinus van n in radialen.
SQR n	Geeft de vierkantswortel van n.
STR$ n	Geeft de string karakters die nodig zouden zijn om de waarde n af te drukken. Voorbeeld: STR$ 256 = "256".
TAB n	Verhoogt de printpositie met n plaatsen.
TAN n	Geeft de tangens van n in radialen.
USR n	Veroorzaakt het aanroepen van een in machinetaal geschreven subroutine, die start op adres n. De waarde van n moet liggen tussen 0 en 65535.
VAL string	Geeft de waarde van de in de string geschreven expressie. Voorbeeld: VAL "3+2*5" = 13 Indien de VAL-functie deel uitmaakt van een uitdrukking, dan moet VAL als eerste in die uitdrukking voorkomen. In PRINT AT, PLOT en UNPLOT statements mag VAL alleen in de eerste coördinaat worden gebruikt.

Beeldschermrooster voor PRINT AT en PLOT

Systeemboodschappen

Het algemeen formaat van systeemboodschappen is:

code/regelnummer

De "code" staat voor een bepaalde boodschap, die betrekking heeft op de met "regelnummer" aangegeven regel. Hierna volgt een overzicht van de mogelijke boodschappen.

code	betekenis	
0		-Normale beëindiging van een goed uitgevoerd programma of beëindiging van een programma door een sprong naar een regelnummer dat hoger is dan enig bestaand regelnummer.
1	NEXT	-Er is geen "control-variabele", maar er is wel een normale variabele die dezelfde naam heeft als de verwachte "control-variabele".
2	NEXT	-Indien er geen "control-variabele" is terwijl er ook geen normale variabele met een zelfde naam is.
	allerlei	-Er wordt een variabele gebruikt die nog niet is toegewezen door middel van een LET- of DIM-statement.
3	allerlei	-Er wordt naar een subscript verwezen met een hoger nummer dan in de DIM-statement werdt gespecificeerd, doch dit nummer ligt wel tussen 0 en 65536.
4	DIM	-Er is niet voldoende geheugenruimte om de gespecificeerde array in te creëren.

code	betekenis	
4	FOR...NEXT	-Er is onvoldoende geheugenruimte voor de controlvariabele.
	GOSUB	-Door het ontbreken van één of meer RETURN-statements wordt de stack steeds groter totdat er geen geheugenruimte meer is om return-adressen op te slaan.
	LIST	-Door het maken van de LIST groeit het beeldscherm geheugen, doch er is onvoldoende ruimte om door te gaan. Met CONT wordt het scherm gewist en kan de LIST worden voortgezet.
	PRINT	-Hiervoor geldt hetzelfde als hierboven bij LIST vermeld staat.
5	AT m,n	-m=22 of m=23. Regels 22 en 23 mogen niet worden beschreven.
	LIST	-Alle regels van het beeldscherm zijn vol. Met behulp van CONT wordt het scherm gewist en gaat LIST door.
—	PRINT	-Alle regels van het beeldscherm zijn vol. Met behulp van CONT wordt het scherm gewist en gaat de PRINT verder.
6	allerlei	-Tijdens een berekening komt "overflow" voor.
7	RETURN	-Er staat geen return-adres in de stack. Dat wil dus zeggen dat er geen bijbehorende GOSUB-statement is geprogrammeerd.
8	INPUT	-INPUT werd als commando ingegeven.
9	STOP	-Het programma is gestopt door het uitvoeren van een STOP-statement.
A	ACS n	-n < -1 of n > +1
	ASN n	-n < -1 of n > +1
	LN n	-n ⩽ 0

code	betekenis	
B	AT m,n	- m < 0 of m > 23; n < 0 of n > 31
	CHR$ n	- n < 0 of n > 255
	DIM A(n,..)	- n ≤ 0 of n > 65535
	PAUSE n	- n < 0 of n > 65535
	PEEK n	- n < 0 of n > 65535
	PLOT m,n	- m < 0 of m > 63; n < 0 of n > 43
	POKE m,n	- m < 0 of m > 65535; n < -255 of n > +255
	RAND n	- n < 0 of n > 65535
	SQR n	- n < 0
	TAB n	- n < 0 of n > 255
	USR n	- n < 0 of n > 65535
C	VAL string	- Er zit een fout in de specificatie van de numerieke variabele die door de string wordt voorgesteld, waardoor de waarde niet kan worden berekend. Er zou bijvoorbeeld een alfanumerieke string in kunnen staan die niets met een numerieke waarde te maken heeft.
D	COPY	- Tijdens het maken van de "copy" is op de BREAK-toets gedrukt.
	INPUT	- Het eerste karakter dat bij de INPUT werd ingegeven was het karakter STOP.
	LLIST	- Tijdens het maken van de listing met LLIST werd op de BREAK-toets gedrukt.
	LOAD	- Tijdens het uitvoeren van het LOAD-commando werd op de BREAK-toets gedrukt, terwijl er nog geen nieuw programma was geladen.
	LPRINT	- Tijdens het printen met LPRINT werd er op de BREAK-toets gedrukt.
	allerlei	- Ook tijdens het uitvoeren van een programmastatement kan men door op de BREAK-toets te drukken de code D veroorzaken. Dit kan bij iedere statement voorkomen.
E	-	- De code E wordt niet gebruikt.
F	SAVE	- Er werd een SAVE-opdracht gegeven waarbij was vergeten een programmanaam op te geven.

Programmeertips

Nieuw geschreven programma's kunnen het beste nog voor ze worden uitgevoerd met behulp van een SAVE-instructie op cassette worden weggeschreven. De kans bestaat namelijk dat een nog niet getest programma tijdens het uitvoeren een zogenaamde systeem "hang up" veroorzaakt, waardoor u genoodzaakt zou zijn de ZX81 aan en uit te schakelen. Dit zou dan weer tot gevolg hebben dat al uw intikwerk voor niets is geweest. Staat uw (niet werkende) programma op cassette, dan bespaart u in ieder geval het opnieuw intikken.

De kans dat er tijdens het wegschrijven van een programma naar cassette een schrijffout wordt gemaakt is relatief groot. Het is daarom ook aan te bevelen ieder programma tenminste twee keer weg te schrijven, liefst naar twee verschillende cassettes. Erg lange programma's kunt u zelfs beter drie keer wegschrijven. Hoe langer een programma, hoe groter de kans dat er een schrijffout wordt gemaakt.

Het is aan te bevelen om programma's standaard te starten met behulp van een GOTO-opdracht. Dit heeft bijna altijd de voorkeur boven het starten met RUN. Met RUN worden namelijk de gegevens die u tijdens een vorige uitvoering van het programma in het geheugen hebt opgeslagen gewist. Alleen indien u alle gegevens uit een vorige uitvoering beslist gewist wilt hebben is het RUN-commando nodig.

Voor het uitvoeren van een GOTO of GOSUB moet de ZX81 het aangegeven regelnummer opzoeken. Dit opzoeken begint steeds bij het begin van het programmageheugen. Het is daarom raadzaam de meest gebruikte subroutines zo ver mogelijk naar het begin van het programma te plaatsen. Dit kan de zoektijd aanzienlijk verkorten, en daarmee de snelheid van het programma aanzienlijk verhogen. Als eerste programmaregel kunt u dan een GOTO plaatsen, waarmee naar het hoofdprogramma wordt gesprongen. Het volgende programma is daar een voorbeeld van:

```
  1Ø GOTO 5ØØ
  2Ø REM SUBROUTINE 1
  3Ø LET A=B
  4Ø LET B=B+1
  5Ø RETURN
 5ØØ REM HOOFDPROGRAMMA
 51Ø LET B=Ø
 52Ø GOSUB 3Ø
 etc.
```

Indien u in uw programma een lus hebt geprogrammeerd waarin een INPUT-statement staat, dan werkt de BREAK-toets vaak niet. Wilt u uw programma toch onderbreken, dan moet u er voor zorgen dat het eerste karakter in de INPUT-string een STOP-code is. Met behulp van de RUBOUT-toets kunt u het eerste aanhalingsteken verwijderen. Daarvoor in de plaats zet u dan een STOP-code. Na NEWLINE zal het programma stoppen met de systeemboodschap "D/(regelnummer).

Een programma dat u van cassette laadt kunt u automatisch laten starten. Hiervoor moet u het programma naar cassette schrijven door middel van een programma statement SAVE. Wanneer SAVE als een programma-instructie wordt gegeven dan wordt namelijk het gehele programma inclusief het beeldschermgeheugen, het variabelengeheugen en een groot aantal systeemvariabelen naar cassette geschreven.
Nadat dit programma van cassette terug in het geheugen is geladen, dan wordt automatisch de eerstvolgende programmaregel na de regel met de SAVE-instructie uitgevoerd.
Wanneer u na de SAVE-instructie een GOTO hebt geprogrammeerd die naar het begin van het programma springt, dan wordt het programma automatisch gestart zonder dat u enig gegeven bent kwijtgeraakt.
Een voorbeeldje van bovenstaande bewering:

```
18ØØ REM EINDE PROGRAMMA
2ØØØ PRINT "ZET RECORDER AAN OP OPNAME"
2Ø1Ø PRINT "RECORDER KLAAR? J/N"
2Ø2Ø INPUT A$
2Ø3Ø IF A$="J" THEN GOTO 2Ø5Ø
2Ø4Ø GOTO 2Ø1Ø
2Ø5Ø SAVE "AUTOSTART"
2Ø6Ø GOTO 1Ø
```

Het automatisch herstarten werkt ook wanneer u met behulp van een rechtstreeks GOTO-commando naar de regel met de SAVE-instructie springt. U moet echter wel bedenken dat door het geven van een rechtstreeks commando de beeldscherm-informatie verloren gaat.

In de 1K uitvoering van de ZX81 is het van belang dat de programma's zo weinig mogelijk geheugenruimte gebruiken. Hier volgen enkele tips om dit te bereiken:

PRINT zoveel mogelijk aan de linkerkant van het beeld-scherm. Hierdoor blijft het beeldschermgeheugen klein.

Probeer tussenresultaten in berekeningen te vermijden. Tussenresultaten ontstaan wanneer in numerieke expressies haakjes worden gebruikt. Vaak kunnen haakjes worden vermeden door de formule op een andere manier op te schrijven.

Gebruik geen gedimensioneerde strings waar dat niet absoluut noodzakelijk is.

Probeer uw programma's zo te stroomlijnen dat er zo weinig mogelijk programmaregels nodig zijn om het gewenste resultaat te bereiken.

Indien u aan meerdere variabelen in de loop van het programma de zelfde waarde wilt toekennen, dan kan het voordeel opleveren aan het begin van het programma een extra regel op te nemen met een waarde toekenning aan een extra variabele. U kunt dan in de andere regels deze variabele gebruiken voor het toekennen van een waarde.

Indien u waarden wilt toekennen aan variabelen of een array wilt definiëren, hetgeen u normaal zou doen aan het begin van het programma, dan kunt u dat ook doen door middel van een rechtstreeks commando. Wel moet u er dan voor waken het programma niet met RUN te starten, maar met een GOTO. Zoals al eerder opgemerkt zou u zich moeten aanwennen de programma's altijd met een GOTO te starten.

Een aantal commando's/instructies hebben elkaar deels overlappende functies met betrekking tot het wissen van delen van het geheugen. Om vergissingen te voorkomen volgen de functies van die commando's/instructies hierna in de vorm van een tabel:

aanschakelen	- Na het aanschakelen van de ZX81 wordt eerst de grootte van het RAM bepaald. De systeemvariabele RAMTOP wordt overeenkomstig gezet. Het gehele RAM (tot RAMTOP) wordt getest en "gewist".
NEW	- Het RAM tot aan het in RAMTOP aangegeven adres wordt getest en gewist (de systeemvariabele RAMTOP wordt niet gewijzigd). Indien RAMTOP met behulp van een POKE-instructie was verlaagd, zou dus alleen het deel van het RAM van het begin tot aan het in RAMTOP aangegeven adres worden getest en gewist. Het deel van RAMTOP tot het werkelijke einde van het RAM wordt niet bewerkt.
CLS	- Het beeldschermgeheugen wordt gewist.
CLEAR	- Het variabelengeheugen wordt gewist.
RUN	- Het variabelengeheugen wordt gewist (CLEAR) en het programma wordt op de eerste regel gestart.
RUN n	- Komt overeen met: CLEAR + GOTO n.
GOTO n	- Er wordt niets gewist. Het programma wordt op de aangegeven regel (n) gestart.

Voor alle rechtstreekse commando's geldt dat ze de momentele inhoud van het beeldschermgeheugen wissen.

PEEK en POKE voorbeelden

In de nu volgende voorbeelden worden PEEK en POKE gebruikt om duidelijk te maken hoe ze toegepast kunnen worden. De voorbeelden zijn zo gekozen dat u ze ook direct kunt gebruiken.

Een waarde uit een geheugenlocatie lezen.

```
PRINT PEEK 17ØØØ
```

Een waarde in een geheugenlocatie schrijven.

```
POKE 17ØØØ,222
```

Het lezen van de systeemvariabele RAMTOP.

```
PRINT PEEK 16389*256+PEEK 16388
```

Het veranderen van RAMTOP.

```
  1Ø LET NEWRAMTOP=17ØØØ
  2Ø POKE 16389,INT (NEWRAMTOP/2
56)
  3Ø POKE 16388,NEWRAMTOP-INT (N
EWRAMTOP/256)*256
```

Na het veranderen van de inhoud van RAMTOP moet een CLS worden gegeven om de nieuwe RAMTOP actief te maken.

Het opvragen van de lengte van het programmageheugen. (D-file) - 16509

```
PRINT (PEEK 16396+PEEK 16397*256)-16509
```

Het opvragen van de lengte van het beeldschermgeheugen. (VARS) - (D-file)

```
PRINT (PEEK 16400+256*PEEK 16401)-(PEEK
16396+256*PEEK 16397)
```

Met behulp van de hiervoor getoonde voorbeelden en de lijst systeemvariabelen kunt u ook allerlei andere gegevens uitlezen of op een bepaalde plaats een waarde schrijven.

USR en machinetaalroutines

Voor het starten van machinetaalprogramma's kunt u gebruik maken van de functie USR nn.
nn is het adres waarop de machinetaalroutine start.
USR is een functie en zal dus altijd bij een instructie moeten worden geplaatst. Enkele voorbeelden:

```
PRINT USR nn
LET A=USR nn
RAND USR nn
```

Wanneer aan het einde van de machinetaalroutine wordt teruggekeerd naar het BASIC-programma, zal het resultaat van de functie USR een geheel getal van twee bytes zijn, zonder teken (+ of -). Dit getal is de inhoud van het registerpaar BC van de Z80.

Dit getal zal dan ook in de instructie worden gebruikt. In de instructie LET A=USR nn zal de variabele A na het uitvoeren van de machinetaalroutine de waarde van het registerpaar BC krijgen. Zo zal de instructie PRINT USR nn, indien BC=0, als resultaat hebben dat er een 0 wordt geprint.

De machinetaalroutine mag geen gebruik maken van de Z80-registers A', F', IX en R.
Dit geldt eigenlijk alleen voor de SLOW-mode, doch om problemen te voorkomen is het beter deze registers nooit te gebruiken.

Bij terugkeer van de machinetaalroutine naar BASIC verwacht het ZX81 systeem dat de registers IY en I de inhoud H'4000' respectievelijk H'1E' hebben. In de machinetaalroutine dient u dan ook vlak voor de 'return' de genoemde registers met die waarden te laden.

Een machinetaalroutine kan op verscheidene plaatsen in het geheugen worden opgeslagen. Er volgen hier vier mogelijkheden. Bij iedere mogelijkheid zijn de voor- en nadelen gegeven.

1. In een REM-instructie die als eerste programmaregel in het BASIC-programma is opgenomen.

Voordeel : Het startadres van de machinetaalroutine is gemakkelijk te vinden, en de machinetaalroutine blijft steeds op dezelfde plaats in het geheugen staan.
Bij een SAVE-opdracht wordt ook de machinetaalroutine naar cassette geschreven.

Nadeel : Niet alle Z80 instructiecodes kunnen in de op deze wijze opgeslagen machinetaalroutine worden gebruikt. NEWLINE kan het BASIC-systeem in de war brengen.

2. In een REM-instructie die als laatste programmaregel in het BASIC-programma is opgenomen.

Voordeel : Bij een SAVE-opdracht wordt ook de machinetaalroutine naar cassette geschreven.

Nadeel : Niet alle Z80 instructiecodes kunnen in de op deze wijze opgeslagen machinetaalroutine worden gebruikt. NEWLINE kan het BASIC-systeem in de war brengen.
Programmawijzigingen verplaatsen ook de laatste REM-instructie, zodat deze methode meer voorzichtigheid vereist bij het bepalen van het startadres van de machinetaalroutine.
Het startadres kan worden bepaald door het begin van het beeldschermgeheugen te nemen en daar de lengte van de machinetaalroutine af te trekken.

3. In het variabelengeheugen (als een string).

Voordeel : Bij een SAVE-opdracht wordt ook de machinetaalroutine naar cassette geschreven.
Alle Z80-instructiecodes kunnen worden gebruikt.

Nadeel : Gegevens in het variabelengeheugen verplaatsen zich continue (vooral in de 1K uitvoering van de ZX81, waar immers ook het beeldschermgeheugen steeds van lengte verandert), waardoor het moeilijk is het startadres van de machinetaalroutine te bepalen.
CLEAR, RUN en NEW mogen niet meer worden gebruikt omdat deze instructies/commando's het variabelen-

geheugen wissen.

4. Boven RAMTOP.

Voordeel : Het startadres van de machinetaalroutine is door u zelf bepaald en verandert niet.
Het geheugendeel boven RAMTOP kan niet door het BASIC-systeem worden beïnvloed. Daar geschreven gegevens blijven in tact zelfs wanneer er een NEW-opdracht wordt gegeven.
Alle Z80-instructiecodes kunnen worden gebruikt.

Nadeel : Bij een SAVE-opdracht wordt de machinetaalroutine niet naar cassette geschreven. Dit nadeel kan alleen worden omzeild door met behulp van PEEK en POKE de machinetaalroutine binnen het BASIC-systeem te copiëren, bijvoorbeeld in een stringvariabele, waarna de SAVE-opdracht kan worden gegeven.
Na een LOAD-opdracht moet de machinetaalroutine eveneens met PEEK en POKE van het BASIC-systeem naar een plaats boven RAMTOP worden verplaatst.

Opmerking:

Indien u met POKE een nieuwe waarde in de systeemvariabele RAMTOP hebt geladen, dan dient u om deze nieuwe waarde aan het BASIC-systeem kenbaar te maken een NEW- of CLS-opdracht te geven.

Bewerkingssymbolen met hun functie

Er zijn drie soorten bewerkingstekens (operators), te weten:

1. rekenkundige operators
2. relatie operators
3. logische operators

1. rekenkundige operators:

**	machtsverheffen
*	vermenigvuldigen
/	delen
+	optellen
-	aftrekken

2. relatie operators:

=	is gelijk aan
>	is groter dan
<	is kleiner dan
<=	is gelijk aan of kleiner dan
>=	is gelijk aan of groter dan
<>	is ongelijk aan

3. logische operators:

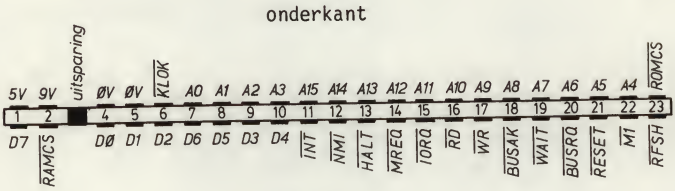
AND	logische EN
OR	logische OF
NOT	logische NIET

Bij het evalueren van bewerkingen in uitdrukkingen hebben die bewerkingen ieder een bepaalde prioriteit. Indien twee bewerkingen dezelfde prioriteit hebben, dan zal de meest linkse bewerking als eerste worden uitgevoerd.
Indien men de prioriteit van bewerkingen wil beïnvloeden, dan kan men dit doen door gebruik te maken van haakjes. De termen tussen de haakjes worden eerst geëvalueerd, geheel volgens de prioriteitsregels. Daarna worden de overblijvende resultaten verder geëvalueerd weer volgens de gegeven prioriteitsregels. In de tabel op de volgende pagina zijn de prioriteiten van de operators gegeven.

signaal	omschrijving	I/O	H/L
$\overline{MREQ}$	memory request (3-status) Geeft aan dat de adresbus een geheugen-adres voor lezen of voor schrijven bevat	O	L
$\overline{IORQ}$	input/output request (3-status) Geeft aan dat de adreslijnen A0 t/m A7 een I/O-device-adres voor lezen of voor schrijven bevatten	O	L
$\overline{RD}$	read (3-status) Geeft aan dat de Z80 wil lezen uit het geheugen of van een I/O-device	O	L
$\overline{WR}$	write (3-status) Geeft aan dat de databuslijnen D0 t/m D7 data bevatten die naar het geheugen of naar een I/O-device moeten worden ge-schreven	O	L
$\overline{RFSH}$	refresh Geeft aan dat de adreslijnen A0 t/m A6 een refresh-adres voor het dynamisch RAM bevatten. De refresh-cyclus kan met $\overline{MREQ}$ worden gestart	O	L
$\overline{HALT}$	stop Geeft aan dat de Z80-CPU zogenaamde NOP-instructies uitvoert, om een refresh-cyclus mogelijk te maken, en dat de Z80 wacht op een interrupt om door te kunnen gaan	O	L
$\overline{WAIT}$	wacht Geeft aan de Z80 te kennen dat het geadresseerde geheugenadres of het I/O-device nog niet direct data kan leveren. De Z80 blijft wachten zolang het signaal $\overline{WAIT}$ actief blijft.	I	L

signaal	omschrijving	I/O	H/L
$\overline{INT}$	interrupt request Hiermee kan een I/O-device de Z80 onderbreken. Indien de Z80 de interrupt accepteert zet het $\overline{IORQ}$	I	L
$\overline{NMI}$	niet te maskeren interrupt Dit is een interrupt request met een hogere prioriteit dan $\overline{INT}$. $\overline{NMI}$ veroorzaakt een restart op adres 0066(hex). Alleen $\overline{BUSRQ}$ heeft een nog hogere prioriteit	I	⌐
$\overline{RESET}$	reset Zet de volgende CPU-registers op nul: PC (program counter) I (interrupt register) R (refresh register) Gedurende het resetten is de CPU niet interruptable. Na de reset begint de CPU met het uitvoeren van de instructie die op adres 0000(hex) staat	I	L
$\overline{BUSRQ}$	bus request De Z80 zal bij het zien van dit signaal alle 3-status signalen op hoge impedantie zetten, om zodoende deze signalen vrij te maken voor gebruik door een ander device (bijvoorbeeld DMA)	I	L
$\overline{BUSAK}$	bus acknowledge Hiermee geeft de Z80 aan dat alle 3-status signalen op hoge impedantie zijn gezet	O	L
KLOK	kloksignaal Het kloksignaal moet van TTL-niveau zijn en heeft een 330 ohm pull-up-weerstand nodig.	I	⊓

ZX 81 parallelpoort met signaalbeschrijving



Op twee na komen alle signalen op de parallel-poort van de ZX81 rechtstreeks van de Z80-CPU. Deze twee niet van de Z80 komende signalen zijn $\overline{\text{ROMCS}}$ en $\overline{\text{RAMCS}}$.
Zie voor een beschrijving van alle andere signalen het hoofdstuk "Z80 CPU met signaalbeschrijvingen".

$\overline{\text{ROMCS}}$ en $\overline{\text{RAMCS}}$ worden door de SCL (Sinclair Computer Logic) gegenereerd uit de adreslijnen A14 en A15.
Ze dienen er voor om extern aangesloten ROM respectievelijk RAM te activeren.

Z 80 registers

De Z80 CPU bevat veertien achtbitsregisters plus vier zestienbitsregisters. De functie van die registers wordt hierna in het kort beschreven.

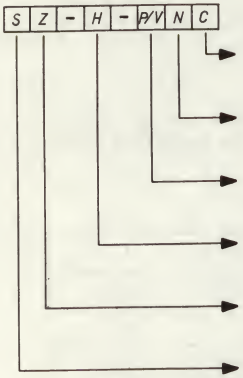
A	*F*	*A'*	*F'*
B	*C*	*B'*	*C'*
D	*E*	*D'*	*E'*
H	*L*	*H'*	*L'*

Register A (accumulator)

De accumulator wordt voor achtbits berekeningen en logische functies gebruikt. Het resultaat van die berekening of die logische bewerking blijft in de accumulator bewaard.

Register F (vlagregister)

Het vlagregister geeft de condities, die het gevolg zijn van acht- of zestienbits bewerkingen, aan.
De functie/betekenis van de bits in dit register worden op de volgende pagina beschreven.



carry; C=1 betekent carry
C=0 betekent geen carry

optel/aftrek; N=0 na optelling
N=1 na aftrekken

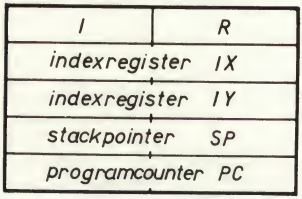
pariteit/overflow; Pariteit voor logische instructies. Overflow voor rekeninstructies.
halve carry; H=1 betekent carry van bit 4 van accu. H=0 betekent geen carry.
zero; Z=1 betekent resultaat van de bewerking is 0. Z=0 betekent resultaat ongelijk aan 0.
sign; S=1 als msb van resultaat een 1 is, anders is S een 0.

Registerparen BC, DE en HL

Deze registerparen kunnen zowel als drie zestienbits-registers alsook als zes achtbits-registers worden gebruikt. Ze zijn vrij te gebruiken door de programmeur.

Alternatieve registers

Deze registers hebben dezelfde functies als de hiervoor beschreven overeenkomstige registers. Met behulp van de exchange-instructies kan de programmeur kiezen uit de door hem gewenste register-set. BC, DE en HL kunnen met één instructie allemaal gewisseld worden met BC', DE' en HL'.



I-register

Indien de Z80 CPU in interruptmode 2 is zal op een interrupt een sprong naar een adres worden uitgevoerd dat is samengesteld uit de acht bits van het I-register (meest significante adresbyte) en acht bits die door het interrumperende randapparaat worden gegenereerd.

R-register

Dit register is een automatische teller voor de "refresh" van dynamisch RAM. Alleen ten behoeve van een test mag de programmeur dit register met een waarde laden, normaliter wordt dit register echter alleen door de Z80 gebruikt.

IX- en IY-registers

In deze indexregisters kan een basisadres worden bewaard die bij geïndexeerd adresseren kan worden gebruikt. Instructies waarmee geïndexeerd kan worden geadresseerd gebruiken naast de waarde uit het indexregister nog een byte voor een verplaatsing ten opzichte van het basisadres.

SP (stack pointer)

In dit register wordt het topadres van de stack bewaard. De stack kan op iedere willekeurige plaats in het geheugen staan (vandaar een 16 bits register) en heeft een zogenaamde LIFO (last in first out) organisatie. Met PUSH zet men data op de stack, met POP haalt men die data er weer af.

PC (program counter)

Hierin bevindt zich het adres van de volgende uit te voeren instructie.

Z 80 instructieset op volgorde van mnemonics

mnemonic		hex.code
ADC	A,(HL)	8E
ADC	A,(IX+d)	DD8E05
ADC	A,(IY+d)	FD8E05
ADC	A,A	8F
ADC	A,B	88
ADC	A,C	89
ADC	A,D	8A
ADC	A,E	8B
ADC	A,H	8C
ADC	A,L	8D
ADC	A,n	CE20
ADC	HL,BC	ED4A
ADC	HL,DE	ED5A
ADC	HL,HL	ED6A
ADC	HL,SP	ED7A
ADD	A,(HL)	86
ADD	A,(IX+d)	DD8605
ADD	A,(IY+d)	FD8605
ADD	A,A	87
ADD	A,B	80
ADD	A,C	81
ADD	A,D	82
ADD	A,E	83
ADD	A,H	84
ADD	A,L	85
ADD	A,n	C620
ADD	HL,BC	09
ADD	HL,DE	19
ADD	HL,HL	29
ADD	HL,SP	39
ADD	IX,BC	DD09
ADD	IX,DE	DD19
ADD	IX,IX	DD29
ADD	IX,SP	DD39
ADD	IY,BC	FD09
ADD	IY,DE	FD19

mnemonic		hex.code
ADD	IY,IY	FD29
ADD	IY,SP	FD39
AND	(HL)	A6
AND	(IX+d)	DDA605
AND	(IY+d)	FDA605
AND	A	A7
AND	B	A0
AND	C	A1
AND	D	A2
AND	E	A3
AND	H	A4
AND	L	A5
AND	n	E620
BIT	0,(HL)	CB46
BIT	0,(IX+d)	DDCB0546
BIT	0,(IY+d)	FDCB0546
BIT	0,A	CB47
BIT	0,B	CB40
BIT	0,C	CB41
BIT	0,D	CB42
BIT	0,E	CB43
BIT	0,H	CB44
BIT	0,L	CB45
BIT	1,(HL)	CB4E
BIT	1,(IX+d)	DDCB054E
BIT	1,(IY+d)	FDCB054E
BIT	1,A	CB4F
BIT	1,B	CB48
BIT	1,C	CB49
BIT	1,D	CB4A
BIT	1,E	CB4B
BIT	1,H	CB4C
BIT	1,L	CB4D
BIT	2,(HL)	CB56
BIT	2,(IX+d)	DDCB0556
BIT	2,(IY+d)	FDCB0556

mnemonic		hex.code	mnemonic		hex.code
BIT	2,A	CB57	BIT	6,E	CB73
BIT	2,B	CB50	BIT	6,H	CB74
BIT	2,C	CB51	BIT	6,L	CB75
BIT	2,D	CB52	BIT	7,(HL)	CB7E
BIT	2,E	CB53	BIT	7,(IX+d)	DDCB057E
BIT	2,H	CB54	BIT	7,(IY+d)	FDCB057E
BIT	2,L	CB55	BIT	7,A	CB7F
BIT	3,(HL)	CB5E	BIT	7,B	CB78
BIT	3,(IX+d)	DDCB055E	BIT	7,C	CB79
BIT	3,(IY+d)	FDCB055E	BIT	7,D	CB7A
BIT	3,A	CB5F	BIT	7,E	CB7B
BIT	3,B	CB58	BIT	7,H	CB7C
BIT	3,C	CB59	BIT	7,L	CB7D
BIT	3,D	CB5A	CALL	C,nn	DC8405
BIT	3,E	CB5B	CALL	M,nn	FC8405
BIT	3,H	CB5C	CALL	NC,nn	D48405
BIT	3,L	CB5D	CALL	NZ,nn	C48405
BIT	4,(HL)	CB66	CALL	P,nn	F48405
BIT	4,(IX+d)	DDCB0566	CALL	PE,nn	EC8405
BIT	4,(IY+d)	FDCB0566	CALL	PO,nn	E48405
BIT	4,A	CB67	CALL	Z,nn	CC8405
BIT	4,B	CB60	CALL	nn	CD8405
BIT	4,C	CB61	CCF		3F
BIT	4,D	CB62	CP	(HL)	BE
BIT	4,E	CB63	CP	(IX+d)	DDBE05
BIT	4,H	CB64	CP	(IY+d)	FDBE05
BIT	4,L	CB65	CP	A	BF
BIT	5,(HL)	CB6E	CP	B	B8
BIT	5,(IX+d)	DDCB056E	CP	C	B9
BIT	5,(IY+d)	FDCB056E	CP	D	BA
BIT	5,A	CB6F	CP	E	BB
BIT	5,B	CB68	CP	H	BC
BIT	5,C	CB69	CP	L	BD
BIT	5,D	CB6A	CP	n	FE20
BIT	5,E	CB6B	CPD		EDA9
BIT	5,H	CB6C	CPDR		ED89
BIT	5,L	CB6D	CPI		EDA1
BIT	6,(HL)	CB76	CPIR		ED81
BIT	6,(IX+d)	DDCB0576	CPL		2F
BIT	6,(IY+d)	FDCB0576	DAA		27
BIT	6,A	CB77	DEC	(HL)	35
BIT	6,B	CB70	DEC	(IX+d)	DD3505
BIT	6,C	CB71	DEC	(IY+d)	FD3505
BIT	6,D	CB72	DEC	A	3D

mnemonic		hex.code
DEC	B	05
DEC	BC	0B
DEC	C	0D
DEC	D	15
DEC	DE	1B
DEC	E	1D
DEC	H	25
DEC	HL	2B
DEC	IX	DD2B
DEC	IY	FD2B
DEC	L	2D
DEC	SP	3B
DI		F3
DJNZ	e	102E
EI		FB
EX	(SP),HL	E3
EX	(SP),IX	DDE3
EX	(SP),IY	FDE3
EX	AF,AF'	08
EX	DE,HL	EB
EXX		D9
HALT		76
IM	0	ED46
IM	1	ED56
IM	2	ED5E
IN	A,(C)	ED78
IN	B,(C)	ED40
IN	C,(C)	ED48
IN	D,(C)	ED50
IN	E,(C)	ED58
IN	H,(C)	ED60
IN	L,(C)	ED68
INC	(HL)	34
INC	(IX+d)	DD3405
INC	(IY+d)	FD3405
INC	A	3C
INC	B	04
INC	BC	03
INC	C	0C
INC	D	14
INC	DE	13
INC	E	1C
INC	H	24
INC	HL	23

mnemonic		hex.code
INC	IX	DD23
INC	IY	FD23
INC	L	2C
INC	SP	33
IN	A,(n)	DB20
IND		EDAA
INDR		EDBA
INI		EDA2
INIR		EDB2
JP	nn	C38405
JP	(HL)	E9
JP	(IX)	DDE9
JP	(IY)	FDE9
JP	C,nn	DA8405
JP	M,nn	FA8405
JP	NC,nn	D28405
JP	NZ,nn	C28405
JP	P,nn	F28405
JP	PE,nn	EA8405
JP	PO,nn	E28405
JP	Z,nn	CA8405
JR	C,e	382E
JR	NC,e	302E
JR	NZ,e	202E
JR	Z,e	282E
JR	e	182E
LD	(BC),A	02
LD	(DE),A	12
LD	(HL),A	77
LD	(HL),B	70
LD	(HL),C	71
LD	(HL),D	72
LD	(HL),E	73
LD	(HL),H	74
LD	(HL),L	75
LD	(HL),n	3620
LD	(IX+d),A	DD7705
LD	(IX+d),B	DD7005
LD	(IX+d),C	DD7105
LD	(IX+d),D	DD7205
LD	(IX+d),E	DD7305
LD	(IX+d),H	DD7405
LD	(IX+d),L	DD7505
LD	(IX+d),n	DD360520

mnemonic		hex.code	mnemonic		hex.code
LD	(IY+d),A	FD7705	LD	C,(HL)	4E
LD	(IY+d),B	FD7005	LD	C,(IX+d)	DD4E05
LD	(IY+d),C	FD7105	LD	C,(IY+d)	FD4E05
LD	(IY+d),D	FD7205	LD	C,A	4F
LD	(IY+d),E	FD7305	LD	C,B	48
LD	(IY+d),H	FD7405	LD	C,C	49
LD	(IY+d),L	FD7505	LD	C,D	4A
LD	(IY+d),n	FD360520	LD	C,E	4B
LD	(nn),A	328405	LD	C,H	4C
LD	(nn),BC	ED438405	LD	C,L	4D
LD	(nn),DE	ED538405	LD	C,n	0E20
LD	(nn),HL	228405	LD	D,(HL)	56
LD	(nn),IX	DD228405	LD	D,(IX+d)	DD5605
LD	(nn),IY	FD228405	LD	D,(IY+d)	FD5605
LD	(nn),SP	ED738405	LD	D,A	57
LD	A,(BC)	0A	LD	D,B	50
LD	A,(DE)	1A	LD	D,C	51
LD	A,(HL)	7E	LD	D,D	52
LD	A,(IX+d)	DD7E05	LD	D,E	53
LD	A,(IY+d)	FD7E05	LD	D,H	54
LD	A,(nn)	3A8405	LD	D,L	55
LD	A,A	7F	LD	D,n	1620
LD	A,B	78	LD	DE,(nn)	ED5B8405
LD	A,C	79	LD	DE,nn	118405
LD	A,D	7A	LD	E,(HL)	5E
LD	A,E	7B	LD	E,(IX+d)	DD5E05
LD	A,H	7C	LD	E,(IY+d)	FD5E05
LD	A,I	ED57	LD	E,A	5F
LD	A,L	7D	LD	E,B	58
LD	A,n	3E20	LD	E,C	59
LD	A,R	ED5F	LD	E,D	5A
LD	B,(HL)	46	LD	E,E	5B
LD	B,(IX+d)	DD4605	LD	E,H	5C
LD	B,(IY+d)	FD4605	LD	E,L	5D
LD	B,A	47	LD	E,n	1E20
LD	B,B	40	LD	H,(HL)	66
LD	B,C	41	LD	H,(IX+d)	DD6605
LD	B,D	42	LD	H,(IY+d)	FD6605
LD	B,E	43	LD	H,A	67
LD	B,H	44	LD	H,B	60
LD	B,L	45	LD	H,C	61
LD	B,n	0620	LD	H,D	62
LD	BC,(nn)	ED4B8405	LD	H,E	63
LD	BC,nn	018405	LD	H,H	64

mnemonic		hex.code
LD	H,L	65
LD	H,n	2620
LD	HL,(nn)	2A8405
LD	HL,nn	218405
LD	I,A	ED47
LD	IX,(nn)	DD2A8405
LD	IX,nn	DD218405
LD	IY,(nn)	FD2A8405
LD	IY,nn	FD218405
LD	L,(HL)	6E
LD	L,(IX+d)	DD6E05
LD	L,(IY+d)	FD6E05
LD	L,A	6F
LD	L,B	68
LD	L,C	69
LD	L,D	6A
LD	L,E	6B
LD	L,H	6C
LD	L,L	6D
LD	L,n	2E20
LD	R,A	ED4F
LD	SP,(nn)	ED7B8405
LD	SP,HL	F9
LD	SP,IX	DDF9
LD	SP,IY	FDF9
LD	SP,nn	318405
LDD		EDA8
LDDR		EDB8
LDI		EDA0
LDIR		EDB0
NEG		ED44
NOP		00
OR	(HL)	B6
OR	(IX+d)	DDB605
OR	(IY+d)	FDB605
OR	A	B7
OR	B	B0
OR	C	B1
OR	D	B2
OR	E	B3
OR	H	B4
OR	L	B5
OR	n	F620
OTDR		ED8B

mnemonic		hex.code
OTIR		EDB3
OUT	(C),A	ED79
OUT	(C),B	ED41
OUT	(C),C	ED49
OUT	(C),D	ED51
OUT	(C),E	ED59
OUT	(C),H	ED61
OUT	(C),L	ED69
OUT	(n),A	D320
OUTD		EDAB
OUTI		EDA3
POP	AF	F1
POP	BC	C1
POP	DE	D1
POP	HL	E1
POP	IX	DDE1
POP	IY	FDE1
PUSH	AF	F5
PUSH	BC	C5
PUSH	DE	D5
PUSH	HL	E5
PUSH	IX	DDE5
PUSH	IY	FDE5
RES	0,(HL)	CB86
RES	0,(IX+d)	DDCB0586
RES	0,(IY+d)	FDCB0586
RES	0,A	CB87
RES	0,B	CB80
RES	0,C	CB81
RES	0,D	CB82
RES	0,E	CB83
RES	0,H	CB84
RES	0,L	CB85
RES	1,(HL)	CB8E
RES	1,(IX+d)	DDCB058E
RES	1,(IY+d)	FDCB058E
RES	1,A	CB8F
RES	1,B	CB88
RES	1,C	CB89
RES	1,D	CB8A
RES	1,E	CB8B
RES	1,H	CB8C
RES	1,L	CB8D
RES	2,(HL)	CB96

mnemonic		hex.code	mnemonic		hex.code
RES	2,(IX+d)	DDCB0596	RES	6,C	CBB1
RES	2,(IY+d)	FDCB0596	RES	6,D	CBB2
RES	2,A	CB97	RES	6,E	CBB3
RES	2,B	CB90	RES	6,H	CBB4
RES	2,C	CB91	RES	6,L	CBB5
RES	2,D	CB92	RES	7,(HL)	CBBE
RES	2,E	CB93	RES	7,(IX+d)	DDCB05BE
RES	2,H	CB94	RES	7,(IY+d)	FDCB05BE
RES	2,L	CB95	RES	7,A	CBBF
RES	3,(HL)	CB9E	RES	7,B	CBB8
RES	3,(IX+d)	DDCB059E	RES	7,C	CBB9
RES	3,(IY+d)	FDCB059E	RES	7,D	CBBA
RES	3,A	CB9F	RES	7,E	CBBB
RES	3,B	CB98	RES	7,H	CBBC
RES	3,C	CB99	RES	7,L	CBBD
RES	3,D	CB9A	RET		C9
RES	3,E	CB9B	RET	C	D8
RES	3,H	CB9C	RET	M	F8
RES	3,L	CB9D	RET	NC	D0
RES	4,(HL)	CBA6	RET	NZ	C0
RES	4,(IX+d)	DDCB05A6	RET	P	F0
RES	4,(IY+d)	FDCB05A6	RET	PE	E8
RES	4,A	CBA7	RET	PO	E0
RES	4,B	CBA0	RET	Z	C8
RES	4,C	CBA1	RETI		ED4D
RES	4,D	CBA2	RETN		ED45
RES	4,E	CBA3	RL	(HL)	CB16
RES	4,H	CBA4	RL	(IX+d)	DDCB0516
RES	4,L	CBA5	RL	(IY+d)	FDCB0516
RES	5,(HL)	CBAE	RL	A	CB17
RES	5,(IX+d)	DDCB05AE	RL	B	CB10
RES	5,(IY+d)	FDCB05AE	RL	C	CB11
RES	5,A	CBAF	RL	D	CB12
RES	5,B	CBA8	RL	E	CB13
RES	5,C	CBA9	RL	H	CB14
RES	5,D	CBAA	RL	L	CB15
RES	5,E	CBAB	RLA		17
RES	5,H	CBAC	RLC	(HL)	CB06
RES	5,L	CBAD	RLC	(IX+d)	DDCB0506
RES	6,(HL)	CBB6	RLC	(IY+d)	FDCB0506
RES	6,(IX+d)	DDCB05B6	RLC	A	CB07
RES	6,(IY+d)	FDCB05B6	RLC	B	CB00
RES	6,A	CBB7	RLC	C	CB01
RES	6,B	CBB0	RLC	D	CB02

mnemonic		hex.code
RLC	E	CB03
RLC	H	CB04
RLC	L	CB05
RLCA		07
RLD		ED6F
RR	(HL)	CB1E
RR	(IX+d)	DDCB051E
RR	(IY+d)	FDCB051E
RR	A	CB1F
RR	B	CB18
RR	C	CB19
RR	D	CB1A
RR	E	CB1B
RR	H	CB1C
RR	L	CB1D
RRA		1F
RRC	(HL)	CB0E
RRC	(IX+d)	DDCB050E
RRC	(IY+d)	FDCB050E
RRC	A	CB0F
RRC	B	CB08
RRC	C	CB09
RRC	D	CB0A
RRC	E	CB0B
RRC	H	CB0C
RRC	L	CB0D
RRCA		0F
RRD		ED67
RST	00H	C7
RST	08H	CF
RST	10H	D7
RST	18H	DF
RST	20H	E7
RST	28H	EF
RST	30H	F7
RST	38H	FF
SBC	A,n	DE20
SBC	A,(HL)	9E
SBC	A,(IX+d)	DD9E05
SBC	A,(IY+d)	FD9E05
SBC	A,A	9F
SBC	A,B	98
SBC	A,C	99
SBC	A,D	9A

mnemonic		hex.code
SBC	A,E	9B
SBC	A,H	9C
SBC	A,L	9D
SBC	HL,BC	ED42
SBC	HL,DE	ED52
SBC	HL,HL	ED62
SBC	HL,SP	ED72
SCF		37
SET	0,(HL)	CBC6
SET	0,(IX+d)	DDCB05C6
SET	0,(IY+d)	FDCB05C6
SET	0,A	CBC7
SET	0,B	CBC0
SET	0,C	CBC1
SET	0,D	CBC2
SET	0,E	CBC3
SET	0,H	CBC4
SET	0,L	CBC5
SET	1,(HL)	CBCE
SET	1,(IX+d)	DDCB05CE
SET	1,(IY+d)	FDCB05CE
SET	1,A	CBCF
SET	1,B	CBC8
SET	1,C	CBC9
SET	1,D	CBCA
SET	1,E	CBCB
SET	1,H	CBCC
SET	1,L	CBCD
SET	2,(HL)	CBD6
SET	2,(IX+d)	DDCB05D6
SET	2,(IY+d)	FDCB05D6
SET	2,A	CBD7
SET	2,B	CBD0
SET	2,C	CBD1
SET	2,D	CBD2
SET	2,E	CBD3
SET	2,H	CBD4
SET	2,L	CBD5
SET	3,(HL)	CBDE
SET	3,(IX+d)	DDCB05DE
SET	3,(IY+d)	FDCB05DE
SET	3,A	CBDF
SET	3,B	CBD8
SET	3,C	CBD9

mnemonic		hex.code
SET	3,D	CBDA
SET	3,E	CBDB
SET	3,H	CBDC
SET	3,L	CBDD
SET	4,(HL)	CBE6
SET	4,(IX+d)	DDCB05E6
SET	4,(IY+d)	FDCB05E6
SET	4,A	CBE7
SET	4,B	CBE0
SET	4,C	CBE1
SET	4,D	CBE2
SET	4,E	CBE3
SET	4,H	CBE4
SET	4,L	CBE5
SET	5,(HL)	CBEE
SET	5,(IX+d)	DDCB05EE
SET	5,(IY+d)	FDCB05EE
SET	5,A	CBEF
SET	5,B	CBE8
SET	5,C	CBE9
SET	5,D	CBEA
SET	5,E	CBEB
SET	5,H	CBEC
SET	5,L	CBED
SET	6,(HL)	CBF6
SET	6,(IX+d)	DDCB05F6
SET	6,(IY+d)	FDCB05F6
SET	6,A	CBF7
SET	6,B	CBF0
SET	6,C	CBF1
SET	6,D	CBF2
SET	6,E	CBF3
SET	6,H	CBF4
SET	6,L	CBF5
SET	7,(HL)	CBFE
SET	7,(IX+d)	DDCB05FE
SET	7,(IY+d)	FDCB05FE
SET	7,A	CBFF
SET	7,B	CBF8
SET	7,C	CBF9
SET	7,D	CBFA
SET	7,E	CBFB
SET	7,H	CBFC
SET	7,L	CBFD

mnemonic		hex.code
SLA	(HL)	CB26
SLA	(IX+d)	DDCB0526
SLA	(IY+d)	FDCB0526
SLA	A	CB27
SLA	B	CB20
SLA	C	CB21
SLA	D	CB22
SLA	E	CB23
SLA	H	CB24
SLA	L	CB25
SRA	(HL)	CB2E
SRA	(IX+d)	DDCB052E
SRA	(IY+d)	FDCB052E
SRA	A	CB2F
SRA	B	CB28
SRA	C	CB29
SRA	D	CB2A
SRA	E	CB2B
SRA	H	CB2C
SRA	L	CB2D
SRL	(HL)	CB3E
SRL	(IX+d)	DDCB053E
SRL	(IY+d)	FDCB053E
SRL	A	CB3F
SRL	B	CB38
SRL	C	CB39
SRL	D	CB3A
SRL	E	CB3B
SRL	H	CB3C
SRL	L	CB3D
SUB	(HL)	96
SUB	(IX+d)	DD9605
SUB	(IY+d)	FD9605
SUB	A	97
SUB	B	90
SUB	C	91
SUB	D	92
SUB	E	93
SUB	H	94
SUB	L	95
SUB	n	D620
XOR	(HL)	AE
XOR	(IX+d)	DDAE05
XOR	(IY+d)	FDAE05

mnemonic		hex.code
XOR	A	AF
XOR	B	A8
XOR	C	A9
XOR	D	AA
XOR	E	AB
XOR	H	AC
XOR	L	AD
XOR	n	EE20

In de hiervoor gegeven tabel voor Z80 instructies op alfabetische volgorde van mnemonic zijn in de hexadecimale codes van de instructies de volgende voorbeeldwaarden opgenomen:

nn - 584H dus in de instructie 8405

d - 5 dus in de instructie 05

n - 20H dus in de instructie 20

e - 2EH dus in de instructie 2E

Dit heeft tot gevolg dat de in deze tabel gegeven instructies een reële lengte hebben. Bij het in machinetaal programmeren behoeft u alleen de door u gewenste waarden voor nn, d, n en e in te vullen.

Z 80 instructieset op volgorde van hexadecimale code

hex.code	mnemonic		hex.code	mnemonic	
00	NOP		23	INC	HL
018405	LD	BC,nn	24	INC	H
02	LD	(BC),A	25	DEC	H
03	INC	BC	2620	LD	H,n
04	INC	B	27	DAA	
05	DEC	B	282E	JR	Z,e
0620	LD	B,n	29	ADD	HL,HL
07	RLCA		2A8405	LD	HL,(nn)
08	EX	AF,AF'	2B	DEC	HL
09	ADD	HL,BC	2C	INC	L
0A	LD	A,(BC)	2D	DEC	L
0B	DEC	BC	2E20	LD	L,n
0C	INC	C	2F	CPL	
0D	DEC	C	302E	JR	NC,e
0E20	LD	C,n	318405	LD	SP,nn
0F	RRCA		328405	LD	(nn),A
102E	DJNZ	e	33	INC	SP
118405	LD	DE,nn	34	INC	(HL)
12	LD	(DE),A	35	DEC	(HL)
13	INC	DE	3620	LD	(HL),n
14	INC	D	37	SCF	
15	DEC	D	382E	JR	C,e
1620	LD	D,n	39	ADD	HL,SP
17	RLA		3A8405	LD	A,(nn)
182E	JR	e	3B	DEC	SP
19	ADD	HL,DE	3C	INC	A
1A	LD	A,(DE)	3D	DEC	A
1B	DEC	DE	3E20	LD	A,n
1C	INC	E	3F	CCF	
1D	DEC	E	40	LD	B,B
1E20	LD	E,n	41	LD	B.C
1F	RRA		42	LD	B,D
202E	JR	NZ,e	43	LD	B,E
218405	LD	HL,nn	44	LD	B,H
228405	LD	(nn),HL	45	LD	B,L

hex.code	mnemonic		hex.code	mnemonic	
46	LD	B,(HL)	72	LD	(HL),D
47	LD	B,A	73	LD	(HL),E
48	LD	C,B	74	LD	(HL),H
49	LD	C,C	75	LD	(HL),L
4A	LD	C,D	76	HALT	
4B	LD	C,E	77	LD	(HL),A
4C	LD	C,H	78	LD	A,B
4D	LD	C,L	79	LD	A,C
4E	LD	C,(HL)	7A	LD	A,D
4F	LD	C,A	7B	LD	A,E
50	LD	D,B	7C	LD	A,H
51	LD	D,C	7D	LD	A,L
52	LD	D,D	7E	LD	A,(HL)
53	LD	D,E	7F	LD	A,A
54	LD	D,H	80	ADD	A,B
55	LD	D,L	81	ADD	A,C
56	LD	D,(HL)	82	ADD	A,D
57	LD	D,A	83	ADD	A,E
58	LD	E,B	84	ADD	A,H
59	LD	E,C	85	ADD	A,L
5A	LD	E,D	86	ADD	A,(HL)
5B	LD	E,E	87	ADD	A,A
5C	LD	E,H	88	ADC	A,B
5D	LD	E,L	89	ADC	A,C
5E	LD	E,(HL)	8A	ADC	A,D
5F	LD	E,A	8B	ADC	A,E
60	LD	H,B	8C	ADC	A,H
61	LD	H,C	8D	ADC	A,L
62	LD	H,D	8E	ADC	A,(HL)
63	LD	H,E	8F	ADC	A,A
64	LD	H,H	90	SUB	B
65	LD	H,L	91	SUB	C
66	LD	H,(HL)	92	SUB	D
67	LD	H,A	93	SUB	E
68	LD	L,B	94	SUB	H
69	LD	L,C	95	SUB	L
6A	LD	L,D	96	SUB	(HL)
6B	LD	L,E	97	SUB	A
6C	LD	L,H	98	SBC	A,B
6D	LD	L,L	99	SBC	A,C
6E	LD	L,(HL)	9A	SBC	A,D
6F	LD	L,A	9B	SBC	A,E
70	LD	(HL),B	9C	SBC	A,H
71	LD	(HL),C	9D	SBC	A,L

hex.code	mnemonic	
9E	SBC	A,(HL)
9F	SBC	A,A
A0	AND	B
A1	AND	C
A2	AND	D
A3	AND	E
A4	AND	H
A5	AND	L
A6	AND	(HL)
A7	AND	A
A8	XOR	B
A9	XOR	C
AA	XOR	D
AB	XOR	E
AC	XOR	H
AD	XOR	L
AE	XOR	(HL)
AF	XOR	A
B0	OR	B
B1	OR	C
B2	OR	D
B3	OR	E
B4	OR	H
B5	OR	L
B6	OR	(HL)
B7	OR	A
B8	CP	B
B9	CP	C
BA	CP	D
BB	CP	E
BC	CP	H
BD	CP	L
BE	CP	(HL)
BF	CP	A
C0	RET	NZ
C1	POP	BC
C28405	JP	NZ,nn
C38405	JP	nn
C48405	CALL	NZ,nn
C5	PUSH	BC
C620	ADD	A,n
C7	RST	0
C8	RET	Z
C9	RET	
CA8405	JP	Z,nn
CB00	RLC	B
CB01	RLC	C
CB02	RLC	D
CB03	RLC	E
CB04	RLC	H
CB05	RLC	L
CB06	RLC	(HL)
CB07	RLC	A
CB08	RRC	B
CB09	RRC	C
CB0A	RRC	D
CB0B	RRC	E
CB0C	RRC	H
CB0D	RRC	L
CB0E	RRC	(HL)
CB0F	RRC	A
CB10	RL	B
CB11	RL	C
CB12	RL	D
CB13	RL	E
CB14	RL	H
CB15	RL	L
CB16	RL	(HL)
CB17	RL	A
CB18	RR	B
CB19	RR	C
CB1A	RR	D
CB1B	RR	E
CB1C	RR	H
CB1D	RR	L
CB1E	RR	(HL)
CB1F	RR	A
CB20	SLA	B
CB21	SLA	C
CB22	SLA	D
CB23	SLA	E
CB24	SLA	H
CB25	SLA	L
CB26	SLA	(HL)
CB27	SLA	A
CB28	SRA	B
CB29	SRA	C
CB2A	SRA	D

hex.code	mnemonic		hex.code	mnemonic	
CB2B	SRA	E	CB5F	BIT	3,A
CB2C	SRA	H	CB60	BIT	4,B
CB2D	SRA	L	CB61	BIT	4,C
CB2E	SRA	(HL)	CB62	BIT	4,D
CB2F	SRA	A	CB63	BIT	4,E
CB38	SRL	B	CB64	BIT	4,H
CB39	SRL	C	CB65	BIT	4,L
CB3A	SRL	D	CB66	BIT	4,(HL)
CB3B	SRL	E	CB67	BIT	4,A
CB3C	SRL	H	CB68	BIT	5,B
CB3D	SRL	L	CB69	BIT	5,C
CB3E	SRL	(HL)	CB6A	BIT	5,D
CB3F	SRL	A	CB6B	BIT	5,E
CB40	BIT	0,B	CB6C	BIT	5,H
CB41	BIT	0,C	CB6D	BIT	5,L
CB42	BIT	0,D	CB6E	BIT	5,(HL)
CB43	BIT	0,E	CB6F	BIT	5,A
CB44	BIT	0,H	CB70	BIT	6,B
CB45	BIT	0,L	CB71	BIT	6,C
CB46	BIT	0,(HL)	CB72	BIT	6,D
CB47	BIT	0,A	CB73	BIT	6,E
CB48	BIT	1,B	CB74	BIT	6,H
CB49	BIT	1,C	CB75	BIT	6,L
CB4A	BIT	1,D	CB76	BIT	6,(HL)
CB4B	BIT	1,E	CB77	BIT	6,A
CB4C	BIT	1,H	CB78	BIT	7,B
CB4D	BIT	1,L	CB79	BIT	7,C
CB4E	BIT	1,(HL)	CB7A	BIT	7,D
CB4F	BIT	1,A	CB7B	BIT	7,E
CB50	BIT	2,B	CB7C	BIT	7,H
CB51	BIT	2,C	CB7D	BIT	7,L
CB52	BIT	2,D	CB7E	BIT	7,(HL)
CB53	BIT	2,E	CB7F	BIT	7,A
CB54	BIT	2,H	CB80	RES	0,B
CB55	BIT	2,L	CB81	RES	0,C
CB56	BIT	2,(HL)	CB82	RES	0,D
CB57	BIT	2,A	CB83	RES	0,E
CB58	BIT	3,B	CB84	RES	0,H
CB59	BIT	3,C	CB85	RES	0,L
CB5A	BIT	3,D	CB86	RES	0,(HL)
CB5B	BIT	3,E	CB87	RES	0,A
CB5C	BIT	3,H	CB88	RES	1,B
CB5D	BIT	3,L	CB89	RES	1,C
CB5E	BIT	3,(HL)	CB8A	RES	1,D

hex.code	mnemonic		hex.code	mnemonic	
CB8B	RES	1,E	CBB7	RES	6,A
CB8C	RES	1,H	CBB8	RES	7,B
CB8D	RES	1,L	CBB9	RES	7,C
CB8E	RES	1,(HL)	CBBA	RES	7,D
CB8F	RES	1,A	CBBB	RES	7,E
CB90	RES	2,B	CBBC	RES	7,H
CB91	RES	2,C	CBBD	RES	7,L
CB92	RES	2,D	CBBE	RES	7,(HL)
CB93	RES	2,E	CBBF	RES	7,A
CB94	RES	2,H	CBC0	SET	0,B
CB95	RES	2,L	CBC1	SET	0,C
CB96	RES	2,(HL)	CBC2	SET	0,D
CB97	RES	2,A	CBC3	SET	0,E
CB98	RES	3,B	CBC4	SET	0,H
CB99	RES	3,C	CBC5	SET	0,L
CB9A	RES	3,D	CBC6	SET	0,(HL)
CB9B	RES	3,E	CBC7	SET	0,A
CB9C	RES	3,H	CBC8	SET	1,B
CB9D	RES	3,L	CBC9	SET	1,C
CB9E	RES	3,(HL)	CBCA	SET	1,D
CB9F	RES	3,A	CBCB	SET	1,E
CBA0	RES	4,B	CBCC	SET	1,H
CBA1	RES	4,C	CBCD	SET	1,L
CBA2	RES	4,D	CBCE	SET	1,(HL)
CBA3	RES	4,E	CBCF	SET	1,A
CBA4	RES	4,H	CBD0	SET	2,B
CBA5	RES	4,L	CBD1	SET	2,C
CBA6	RES	4,(HL)	CBD2	SET	2,D
CBA7	RES	4,A	CBD3	SET	2,E
CBA8	RES	5,B	CBD4	SET	2,H
CBA9	RES	5,C	CBD5	SET	2,L
CBAA	RES	5,D	CBD6	SET	2,(HL)
CBAB	RES	5,E	CBD7	SET	2,A
CBAC	RES	5,H	CBD8	SET	3,B
CBAD	RES	5,L	CBD9	SET	3,C
CBAE	RES	5,(HL)	CBDA	SET	3,D
CBAF	RES	5,A	CBDB	SET	3,E
CBB0	RES	6,B	CBDC	SET	3,H
CBB1	RES	6,C	CBDD	SET	3,L
CBB2	RES	6,D	CBDE	SET	3,(HL)
CBB3	RES	6,E	CBDF	SET	3,A
CBB4	RES	6,H	CBE0	SET	4,B
CBB5	RES	6,L	CBE1	SET	4,C
CBB6	RES	6,(HL)	CBE2	SET	4,D

hex.code	mnemonic		hex.code	mnemonic	
CBE3	SET	4,E	DB20	IN	A,n
CBE4	SET	4,H	DC8405	CALL	C,nn
CBE5	SET	4,L	DD09	ADD	IX,BC
CBE6	SET	4,(HL)	DD19	ADD	IX,DE
CBE7	SET	4,A	DD218405	LD	IX,nn
CBE8	SET	5,B	DD228405	LD	(nn),IX
CBE9	SET	5,C	DD23	INC	IX
CBEA	SET	5,D	DD29	ADD	IX,IX
CBEB	SET	5,E	DD2A8405	LD	IX,(nn)
CBEC	SET	5,H	DD2B	DEC	IX
CBED	SET	5,L	DD3405	INC	(IX+d)
CBEE	SET	5,(HL)	DD3505	DEC	(IX+d)
CBEF	SET	5,A	DD360520	LD	(IX+d),n
CBF0	SET	6,B	DD39	ADD	IX,SP
CBF1	SET	6,C	DD4605	LD	B,(IX+d)
CBF2	SET	6,D	DD4E05	LD	C,(IX+d)
CBF3	SET	6,E	DD5605	LD	D,(IX+d)
CBF4	SET	6,H	DD5E05	LD	E,(IX+d)
CBF5	SET	6,L	DD6605	LD	H,(IX+d)
CBF6	SET	6,(HL)	DD6E05	LD	L,(IX+d)
CBF7	SET	6,A	DD7005	LD	(IX+d),B
CBF8	SET	7,B	DD7105	LD	(IX+d),C
CBF9	SET	7,C	DD7205	LD	(IX+d),D
CBFA	SET	7,D	DD7305	LD	(IX+d),E
CBFB	SET	7,E	DD7405	LD	(IX+d),H
CBFC	SET	7,H	DD7505	LD	(IX+d),L
CBFD	SET	7,L	DD7705	LD	(IX+d),A
CBFE	SET	7,(HL)	DD7E05	LD	A,(IX+d)
CBFF	SET	7,A	DD8605	ADD	A,(IX+d)
CC8405	CALL	Z,nn	DD8E05	ADC	A,(IX+d)
CD8405	CALL	nn	DD9605	SUB	(IX+d)
CE20	ADC	A,n	DD9E05	SBC	A,(IX+d)
CF	RST	8	DDA605	AND	(IX+d)
D0	RET	NC	DDAE05	XOR	(IX+d)
D1	POP	DE	DDB605	OR	(IX+d)
D28405	JP	NC,nn	DDBE05	CP	(IX+d)
D320	OUT	n,A	DDCB0506	RLC	(IX+d)
D48405	CALL	NC,nn	DDCB050E	RRC	(IX+d)
D5	PUSH	DE	DDCB0516	RL	(IX+d)
D620	SUB	n	DDCB051E	RR	(IX+d)
D7	RST	10	DDCB0526	SLA	(IX+d)
D8	RET	C	DDCB052E	SRA	(IX+d)
D9	EXX		DDCB053E	SRL	(IX+d)
DA8405	JP	C,nn	DDCB0546	BIT	0,(IX+d)

hex.code	mnemonic		hex.code	mnemonic	
DDCB054E	BIT	1,(IX+d)	ED41	OUT	(C),B
DDCB0556	BIT	2,(IX+d)	ED42	SBC	HL,BC
DDCB055E	BIT	3,(IX+d)	ED438405	LD	(nn),BC
DDCB0566	BIT	4,(IX+d)	ED44	NEG	
DDCB056E	BIT	5,(IX+d)	ED45	RETN	
DDCB0576	BIT	6,(IX+d)	ED46	IM	0
DDCB057E	BIT	7,(IX+d)	ED47	LD	I,A
DDCB0586	RES	0,(IX+d)	ED48	IN	C,(C)
DDCB058E	RES	1,(IX+d)	ED49	OUT	(C),C
DDCB0596	RES	2,(IX+d)	ED4A	ADC	HL,BC
DDCB059E	RES	3,(IX+d)	ED4B8405	LD	BC,(nn)
DDCB05A6	RES	4,(IX+d)	ED4D	RETI	
DDCB05AE	RES	5,(IX+d)	ED50	IN	D,(C)
DDCB05B6	RES	6,(IX+d)	ED51	OUT	(C),D
DDCB05BE	RES	7,(IX+d)	ED52	SBC	HL,DE
DDCB05C6	SET	0,(IX+d)	ED538405	LD	(nn),DE
DDCB05CE	SET	1,(IX+d)	ED56	IM	1
DDCB05D6	SET	2,(IX+d)	ED57	LD	A,I
DDCB05DE	SET	3,(IX+d)	ED58	IN	E,(C)
DDCB05E6	SET	4,(IX+d)	ED59	OUT	(C),E
DDCB05EE	SET	5,(IX+d)	ED5A	ADC	HL,DE
DDCB05F6	SET	6,(IX+d)	ED5B8405	LD	DE,(nn)
DDCB05FE	SET	7,(IX+d)	ED5E	IM	2
DDE1	POP	IX	ED60	IN	H,(C)
DDE3	EX	(SP),IX	ED61	OUT	(C),H
DDE5	PUSH	IX	ED62	SBC	HL,HL
DDE9	JP	(IX)	ED67	RRD	
DDF9	LD	SP,IX	ED68	IN	L,(C)
DE20	SBC	A,n	ED69	OUT	(C),L
DF	RST	18	ED6A	ADC	HL,HL
E0	RET	PO	ED6F	RLD	
E1	POP	HL	ED72	SBC	HL,SP
E28405	JP	PO,nn	ED738405	LD	(nn),SP
E3	EX	(SP),HL	ED78	IN	A,(C)
E48405	CALL	PO,nn	ED79	OUT	(C),A
E5	PUSH	HL	ED7A	ADC	HL,SP
E620	AND	n	ED7B8405	LD	SP,(nn)
E7	RST	20	EDA0	LDI	
E8	RET	PE	EDA1	CPI	
E9	JP	(HL)	EDA2	INI	
EA8405	JP	PE,nn	EDA3	OUTI	
EB	EX	DE,HL	EDA8	LDD	
EC8405	CALL	PE,nn	EDA9	CPD	
ED40	IN	B,(C)	EDAA	IND	

hex.code	mnemonic		hex.code	mnemonic	
EDAB	OUTD		FD7205	LD	(IY+d),D
EDB0	LDIR		FD7305	LD	(IY+d),E
EDB1	CPIR		FD7405	LD	(IY+d),H
EDB2	INIR		FD7505	LD	(IY+d),L
EDB3	OTIR		FD7705	LD	(IY+d),A
EDB8	LDDR		FD7E05	LD	A,(IY+d)
EDB9	CPDR		FD8605	ADD	A,(IY+d)
EDBA	INDR		FD8E05	ADC	A,(IY+d)
EDBB	OTDR		FD9605	SUB	(IY+d)
EE20	XOR	n	FD9E05	SBC	A,(IY+d)
EF	RST	28	FDA605	AND	(IY+d)
F0	RET	P	FDAE05	XOR	(IY+d)
F1	POP	AF	FDB605	OR	(IY+d)
F28405	JP	P,nn	FDBE05	CP	(IY+d)
F3	DI		FDCB0506	RLC	(IY+d)
F48405	CALL	P,nn	FDCB050E	RRC	(IY+d)
F5	PUSH	AF	FDCB0516	RL	(IY+d)
F620	OR	n	FDCB051E	RR	(IY+d)
F7	RST	30	FDCB0526	SLA	(IY+d)
F8	RET	M	FDCB052E	SRA	(IY+d)
F9	LD	SP,HL	FDCB053E	SRL	(IY+d)
FA8405	JP	M,nn	FDCB0546	BIT	0,(IY+d)
FB	EI		FDCB054E	BIT	1,(IY+d)
FC8405	CALL	M,nn	FDCB0556	BIT	2,(IY+d)
FD09	ADD	IY,BC	FDCB055E	BIT	3,(IY+d)
FD19	ADD	IY,DE	FDCB0566	BIT	4,(IY+d)
FD218405	LD	IY,nn	FDCB056E	BIT	5,(IY+d)
FD228405	LD	(nn),IY	FDCB0576	BIT	6,(IY+d)
FD23	INC	IY	FDCB057E	BIT	7,(IY+d)
FD29	ADD	IY,IY	FDCB0586	RES	0,(IY+d)
FD2A8405	LD	IY,(nn)	FDCB058E	RES	1,(IY+d)
FD2B	DEC	IY	FDCB0596	RES	2,(IY+d)
FD3405	INC	(IY+d)	FDCB059E	RES	3,(IY+d)
FD3505	DEC	(IY+d)	FDCB05A6	RES	4,(IY+d)
FD360520	LD	(IY+d),n	FDCB05AE	RES	5,(IY+d)
FD39	ADD	IY,SP	FDCB05B6	RES	6,(IY+d)
FD4605	LD	B,(IY+d)	FDCB05BE	RES	7,(IY+d)
FD4E05	LD	C,(IY+d)	FDCB05C6	SET	0,(IY+d)
FD5605	LD	D,(IY+d)	FDCB05CE	SET	1,(IY+d)
FD5E05	LD	E,(IY+d)	FDCB05D6	SET	2,(IY+d)
FD6605	LD	H,(IY+d)	FDCB05DE	SET	3,(IY+d)
FD6E05	LD	L,(IY+d)	FDCB05E6	SET	4,(IY+d)
FD7005	LD	(IY+d),B	FDCB05EE	SET	5,(IY+d)
FD7105	LD	(IY+d),C	FDCB05F6	SET	6,(IY+d)

hex.code	mnemonic	
FDCB05FE	SET	7,(IY+d)
FDE1	POP	IY
FDE3	EX	(SP),IY
FDE5	PUSH	IY
FDE9	JP	(IY)
FDF9	LD	SP,IY
FE20	CP	n
FF	RST	38

In de hiervoor gegeven tabel voor Z80 instructies op volgorde van hexadecimale code zijn in de hexadecimale codes van de instructies de volgende voorbeeldwaarden opgenomen:

nn - 584H dus in de instructie 8405

d - 5 dus in de instructie 05

n - 20H dus in de instructie 20

e - 2EH dus in de instructie 2E

Dit heeft tot gevolg dat de in deze tabel opgenomen instructies een reële lengte hebben. Bij het uitlezen van een stuk machinetaalprogramma zult u echter andere waarden tegenkomen.

Z 80 vlagbeïnvloeding

instructies	C	Z	P/V	S	N	H	opmerkingen
ADD A,s; ADC A,s	A	A	V	A	0	A	8 bits optelinstructie
SUB s; SBC A,s; CP s	A	A	V	A	1	A	8 bits aftrekinstructies, vergelijk- en
NEG							negatie-instructies.
AND s	0	A	P	A	0	1	} logische
OR s; XOR s	0	A	P	A	0	0	} bewerkingen
INC s	C	A	V	A	0	A	8 bits increment
DEC m	C	A	V	A	1	A	8 bits decrement
ADD dd,ss	A	C	C	C	0	X	16 bits optellen
ADC HL,ss	A	A	V	A	0	X	16 bits optellen+carry
SBC HL,ss	A	A	V	A	1	X	16 bits aftrek + carry
RLA; RLCA; RRA; RRCA	A	C	C	C	0	0	roteren accumulator
RL m; RLC m; RR m;	A	A	P	A	0	0	roteren en schuiven
RRC m; SLA m; SRA m;							van locatie m
SRL m							
RLD; RRD	C	A	P	A	0	0	roteren tetrade links en rechts.
DAA	A	A	P	A	C	A	decimal adjust accu.
CPL	C	C	C	C	1	1	complementeren accu.
SCF	1	C	C	C	0	0	zet carry-vlag
CCF	A	C	C	C	0	X	complement. carry-vlag
IN r,(C)	C	A	P	A	0	0	input register indirect
INI; IND; OUTI; OUTD;	C	A	X	X	1	X	} blok-I/O. Indien B≠0
INIR; INDR; OTIR; OTDR	C	1	X	X	1	X	} dan Z=0 anders Z=1
LDI; LDD	C	X	A	X	0	0	} blok-transfer. Indien
LDIR; LDDR	C	X	0	X	0	0	} BC≠0 dan P/V=1, anders P/V=0
CPI; CPIR; CPD; CPDR	C	A	A	A	1	X	blokzoekinstructies. Indien A=(HL) dan Z=1, anders Z=0. Indien BC≠0 dan P/V=1, anders P/V=0

instructies	C	Z	P/V	S	N	H	opmerkingen
LD A,I; LD A,R	C	A	F	A	0	0	inhoud van Interrupt flipflop (IFF) naar P/V vlag.
BIT b,s	C	A	X	X	0	1	Het complement van bit b uit locatie s naar Z-vlag.

A = Vlag wordt afhankelijk van het resultaat van de bewerking gezet.
C = Vlag blijft onveranderd.
0 = Vlag wordt ge-reset.
1 = Vlag wordt ge-set.
X = Status van de vlag is niet van belang.
V = De "overflow"-vlag wordt al naar gelang het resultaat van de bewerking ge-set.
P = De pariteitsvlag wordt al naar gelang het resultaat van de bewerking ge-set.
r = Eén van de CPU-registers A, B, C, D, E, H of L.
s = 8 bits geheugenlocatie.
ss = 16 bits geheugenlocatie.
m = 8 bits geheugenlocatie.
n = 8 bits waarde (0 t/m 255)
nn = 16 bits waarde (0 t/m 65535)

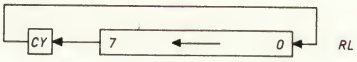
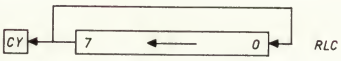
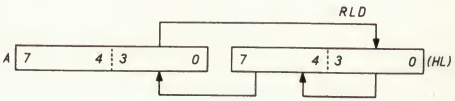
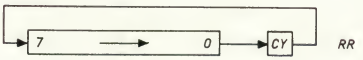
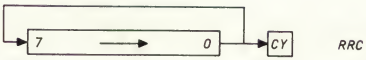
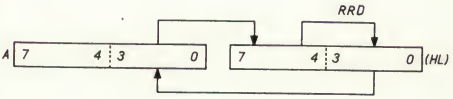
Symbolische omschrijving van Z 80 machine-instructies

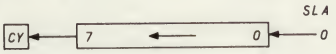
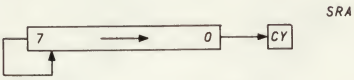
In de hiernavolgende tabel zijn op alfabetische volgorde van mnemonic een aantal aantekeningen opgenomen. Alleen die instructies die meer doen dan hun mnemonic doet vermoeden staan in de tabel.

mnemonic	symbolische omschrijving
BIT b,t	$Z \leftarrow \overline{t_b}$ (test bit b op locatie t)
CALL cc,nn	Indien cc=waar dan doorgaan, anders:
	$(SP-1) \leftarrow PC_H$
	$(SP-2) \leftarrow PC_L$
	$PC \leftarrow nn$
CALL nn	$(SP-1) \leftarrow PC_H$
	$(SP-2) \leftarrow PC_L$
	$PC \leftarrow nn$
CCF	$CY \leftarrow \overline{CY}$
CP s	A-s (A blijft ongewijzigd. Alleen de vlaggen worden gezet)
CPD	A-(HL)
	HL$\leftarrow$ HL-1
	BC$\leftarrow$ BC-1
CPDR	Als CPD doch wordt herhaald tot A=(HL) of tot BC=0
CPI	A-(HL)
	HL$\leftarrow$ HL+1
	BC$\leftarrow$ BC-1

mnemonic	symbolische omschrijving
CPIR	Als CPI doch wordt herhaald tot A=(HL) of tot BC=0
CPL	$A \leftarrow \overline{A}$
DAA	Converteert de inhoud van A naar "packed BCD" indien volgend op een optel- of aftrekinstructie met "packed BCD"-operand
DI	IFF ← 0 (IFF = interrupt flipflop)
DJNZ e	B ← B-1 Indien B=0 dan doorgaan. Indien B≠0 dan PC ← PC+e.
EI	IFF ← 1 (IFF = interrupt flipflop)
EX (SP),HL	H ← (SP+1) L ← (SP)
EX (SP),IX	$IX_H \leftarrow$ (SP+1) $IX_L \leftarrow$ (SP)
IN r,(C)	r ← (C) Indien r = 110 dan worden alleen vlaggen gezet.
IND	(HL) ← (C) B ← B-1 HL ← HL-1
INDR	Als IND doch wordt herhaald tot B=0.
INI	(HL) ← (C) B ← B-1 HL ← HL+1
INIR	Als INI doch wordt herhaald tot B=0.
JP nn	PC ← nn
JP cc,nn	Indien cc = waar dan PC ← nn anders doorgaan,

mnemonic	symbolische omschrijving
JR e	PC ← PC+e
JR cond,e	Indien conditie = waar doorgaan, anders
	PC ← PC+e
	(cond = C, NC, Z of NZ)
LD (nn),dd	(nn+1) ← dd_H
	(nn) ← dd_L
LD dd,(nn)	dd_H ← (nn+1)
	dd_L ← (nn)
LDD	(DE) ← (HL)
	DE ← DE-1
	HL ← HL-1
	BC ← BC-1
LDDR	Als LDD doch wordt herhaald tot BC=0.
LDI	(DE) ← (HL)
	DE ← DE+1
	HL ← HL+1
	BC ← BC-1
LDIR	Als LDI doch wordt herhaald tot BC=0.
NEG	A ← 0-A
OUTD	(C) ← (HL)
	B ← B-1
	HL ← HL-1
OTDR	Als OUTD doch wordt herhaald tot B=0.
OUTI	(C) ← (HL)
	B ← B-1
	HL ← HL+1
OTIR	Als OUTI doch wordt herhaald tot B=0.
POP qq	qq_H ← (SP+1)
	qq_L ← (SP)

mnemonic	symbolische omschrijving
PUSH qq	$(SP-2) \leftarrow qq_L$ $(SP-1) \leftarrow qq_H$
RET	$PC_L \leftarrow (SP)$ $PC_H \leftarrow (SP+1)$
RET cc	Indien cc = waar dan als RET.
RL/RLA	
RLC/RLCA	
RLD	
RR/RRA	
RRC/RRCA	
RRD	

mnemonic	symbolische omschrijving
RST p	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ $PC_H \leftarrow 0$ $PC_L \leftarrow p$
SCF	$CY \leftarrow 1$
SLA	
SRA	
SRL	

In de voorgaande tabel komen een aantal symbolen voor waarvan hierna de betekenis wordt gegeven:

cc	conditie	dd	p	qq	s
000	NZ non zero	BC	00H	BC	r
001	Z zero	DE	08H	DE	n
010	NC not carry	HL	10H	HL	(HL)
011	C carry	IX	18H	AF	(IX+d)
100	PO pariteit oneven	IY	20H	IX	(IY+d)
101	PE pariteit even	SP	28H	IY	
110	P positief		30H		
111	M negatief		38H		

Programmeer-formulier voor machinetaalprogramma's

programma: *datum:* - - . *paginanummer:*

adres dec.	label	instructie: symbolisch opcode	operands	adres hex.	instructie: hex.code 1	2	3	4	opmerkingen

Decimaal geheugendumpprogramma

Dit programma maakt een geheugendump, waarbij de inhoud van iedere byte in de decimale vorm wordt afgedrukt.
U wordt eerst gevraagd op te geven vanaf welk adres u de geheugendump wilt hebben. Daarna zal het programma steeds in blokken van 64 bytes een deel van het geheugen op het beeldscherm afdrukken. Na het afdrukken van een blok kunt u het programma het volgende blok van 64 bytes laten afdrukken door op de vraag "DOORGAAN?" met J te antwoorden. Iedere regel van de output begint met het adres van de byte uit de eerste kolom van de vier kolommen die de inhoud van het geheugen weergeven. Het formaat van de regels is dus:

```
adres    dec    dec    dec    dec
```

```
  1 REM *** DEC. GEHEUGENDUMP
  2 REM *** COPYRIGHT 1983
 10 PRINT "STARTADRES?"
 20 INPUT S
 30 CLS
 40 FOR B=0 TO 15
 50 PRINT S;
 60 FOR R=0 TO 3
 70 PRINT AT B,R*4+6;PEEK (S+R)
 80 NEXT R
 90 LET S=S+4
100 NEXT B
110 PRINT "DOORGAAN? J/N"
120 INPUT D$
130 IF D$="J" THEN GOTO 30
140 STOP
```

Hexadecimaal geheugendumpprogramma

Vanaf het door u opgegeven startadres maakt dit programma een geheugendump van de gelezen bytes, waarbij de inhoud van die bytes in hexadecimaal formaat op het beeldscherm worden afgedrukt.
Er worden steeds blokken van 64 bytes afgedrukt, waarna u wordt gevraagd of u wilt doorgaan of niet.
De op het scherm af te drukken regels hebben het volgende formaat:

adres hex hex hex hex

```
  1 REM *** HEX. GEHEUGENDUMP
  2 REM *** COPYRIGHT 1983
 10 DIM H$(2)
 20 PRINT "STARTADRES?"
 30 INPUT S
 40 CLS
 50 FOR B=0 TO 15
 60 PRINT S;
 70 FOR R=0 TO 3
 80 LET C=PEEK (S+R)
 90 LET D=C
100 FOR I=2 TO 1 STEP -1
110 LET C=INT (D/16)
120 LET H$(I)=CHR$ (D-16*C+28)
130 LET D=C
140 NEXT I
150 PRINT AT B,R*3+6;H$(1);H$(2)
160 NEXT R
170 LET S=S+4
180 NEXT B
190 PRINT "DOORGAAN? J/N"
200 INPUT D$
210 IF D$="J" THEN GOTO 40
220 STOP
```

Karakter geheugendumpprogramma

Dit programma leest de door u opgegeven geheugenlocaties en drukt vervolgens de karakters af die bij de gelezen codes hoort.
Het geheugen wordt in blokken van 64 bytes gelezen, waarna u op de vraag "DOORGAAN?" door met "J" te antwoorden een volgend blok kunt laten afdrukken.
Het eerste byte van het eerste blok kunt u opgeven door op de vraag "STARTADRES?" het door u gewenste adres in decimale vorm op te geven.
De door het programma te genereren output op het scherm heeft het volgende formaat:

adres kar kar kar kar

```
  1 REM *** KAR. GEHEUGENDUMP
  2 REM *** COPYRIGHT 1983
 10 PRINT "STARTADRES?"
 20 INPUT S
 30 CLS
 40 FOR B=0 TO 15
 50 PRINT S;
 60 FOR R=0 TO 3
 70 PRINT AT B,R*6+6;CHR$ PEEK (S+R)
 80 NEXT R
 90 LET S=S+4
100 NEXT B
110 PRINT "DOORGAAN? J/N"
120 INPUT D$
130 IF D$="J" THEN GOTO 30
140 STOP
```

Decimaal, hexadecimaal en karakter geheugendumpprogramma

Vanaf het door u opgegeven startadres maakt dit programma een dump van het geheugen. De gelezen bytes worden op het beeldscherm afgedrukt in zowel decimaal, hexadecimaal als karakter formaat.
Het geheugen wordt in blokken van 16 bytes op het scherm afgedrukt. U kunt een volgend blok laten afdrukken door op de vraag "DOORGAAN?" met "J" te antwoorden.
De op het beeldscherm af te drukken regels hebben het volgende formaat:

```
adres    dec    hex    kar
```

```
  1 REM *** DECHEXCHAR-DUMP
  2 REM *** COPYRIGHT 1983
 10 DIM H$(2)
 20 PRINT "STARTADRES?"
 30 INPUT S
 40 CLS
 50 FOR A=0 TO 15
 60 PRINT S;
 70 LET B=PEEK S
 80 PRINT AT A,6;B
 90 LET C=B
100 FOR I=2 TO 1 STEP -1
110 LET B=INT (C/16)
120 LET H$(I)=CHR$ (C-16*B+28)
130 LET C=B
140 NEXT I
150 PRINT AT A,10;H$(1);H$(2)
160 PRINT AT A,14;CHR$ PEEK S
170 LET S=S+1
180 NEXT A
190 PRINT "DOORGAAN? J/N"
200 INPUT D$
210 IF D$="J" THEN GOTO 40
220 STOP
```

ROM-testprogramma

De BASIC-ROM bevat een groot aantal routines die niet allemaal even geregeld gebruikt worden. Het is dan ook mogelijk dat een groot aantal programma's foutloos draaien, terwijl er één programma is dat in de fout gaat. Maar al te gemakkelijk zou men kunnen denken dat er een fout in het programma zit. Kan men dan ook geen fout in dat programma vinden, dan zou het aanbeveling verdienen de BASIC-ROM eens te testen.
Het volgende programma voert zo'n test uit. De test is erg eenvoudig. De ROM bevat 8K bytes. Het programma telt gewoon deze 8192 bytes bij elkaar op. Aan het eind van het programma wordt het resultaat gecontroleerd en wordt er een bericht op het scherm geschreven, waaruit blijkt of de ROM correct is (er wordt nog onderscheid gemaakt tussen de oude en de nieuwe ZX81-ROM), of dat de ROM fout is.
Daar er in dit programma erg veel gerekend wordt, er moeten immers 8192 optellingen worden verricht, wordt dit programma in de FAST-mode uitgevoerd. U zult dus na het starten het scherm zwart zien worden. Na ongeveer 60 seconden moet dan het bericht omtrent de conditie van de ROM op het scherm verschijnen.

```
   1 REM *** ROMTEST ZX81
   2 REM *** COPYRIGHT 1983
  10 FAST
  20 LET TOTAAL=0
  30 FOR A=0 TO 8191
  40 LET TOTAAL=TOTAAL+PEEK A
  50 NEXT A
  60 IF TOTAAL=855106 THEN PRINT
 "NIEUWE ROM ZX81 / OK"
  70 IF TOTAAL=854885 THEN PRINT
 "OUDE ROM ZX81 / OK"
  80 IF TOTAAL<>855106 AND TOTAA
L<>854885 THEN PRINT "ROM FOUT"
  90 STOP
```

Programma voor het laden van hexadecimale code boven RAMTOP

Dit programma kunt u gebruiken voor het ingeven van machinetaal-programma's (hexadecimale codes) tussen RAMTOP en het einde van het geheugen. Normaal staat het adres van het einde van het geheugen in de systeemvariabele RAMTOP. Met dit programma wordt RAMTOP op een door u gewenste waarde gezet.
Op de vraag "ADRES" kunt u het adres ingeven waarop uw machinetaalprogramma moet beginnen. RAMTOP wordt automatisch ook op dit adres gezet, zodat uw machinetaalprogramma niet meer door het BASIC-systeem kan worden overschreven.
Het nadeel van het plaatsen van instructies boven RAMTOP is dat deze instructies niet op cassette worden geschreven bij een SAVE-commando.
Op de vraag "AANTAL BYTES" kunt u opgeven hoeveel bytes u wilt gaan ingeven. Zodra dat aantal bereikt is wordt het programma beëindigd. Wilt u minder bytes ingeven dan u had opgegeven, dan kunt u in plaats van een hexadecimale waarde de letters "SS" ingeven. Hierdoor stopt het programma verdere input.
Op uw beeldscherm verschijnt nu de tekst "HEXCODE" met daarachter een adres. Dit is het adres waar het volgende door u in te geven byte zal worden weggeschreven.
Indien u per ongeluk een niet-hexadecimale waarde intikt, dan zult u zien dat het adres niet wordt opgehoogd. U kunt alsnog een hexadecimale waarde ingeven.
Indien u dit programma wilt gebruiken op een 1K ZX81, dan moet u de programmaregels 1 en 2 weglaten, daar u anders te weinig geheugen overhoud tijdens het uitvoeren van het programma.

De listing van het programma is op de volgende pagina weergegeven.

```
  1 REM *** HEXLADER
  2 REM *** COPYRIGHT 1983
 10 DIM H$(2)
 20 PRINT "ADRES"
 30 INPUT S
 40 POKE 16389,INT (S/256)
 50 POKE 16388,S-INT (S/256)*25
6
 60 PRINT"AANTAL BYTES"
 70 INPUT B
 80 CLS
 90 FOR A=S TO S+(B-1)
100 PRINT AT 0,0;"HEXCODE ";A
110 INPUT H$
120 LET M=CODE H$(1)-28
130 LET L=CODE H$(2)-28
140 IF H$="SS" THEN GOTO 180
150 IF M>15 OR L>15 THEN GOTO 1
10
160 POKE A,M*16+L
170 NEXT A
180 STOP
```

Aantekeningen

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

Voor de beginnende-, de gevorderde-, de BASIC- en de machinetaalprogrammeur zijn nu alle belangrijke gegevens over de ZX 81 samengevat in dit handige zakboekje.
De gegevens zijn in de volgende vier hoofdgroepen samengebracht:

a. Algemeen
b. BASIC-systeem
c. Hardware
d. Machinetaal

Voor het uitlezen van het geheugen, het testen van de ROM en het laden van machinetaalprogramma's zijn een aantal kant en klare programma's opgenomen, die allemaal in de 1 K versie van de ZX 81 kunnen worden geladen en uitgevoerd.

ISBN 90 201 1639 8